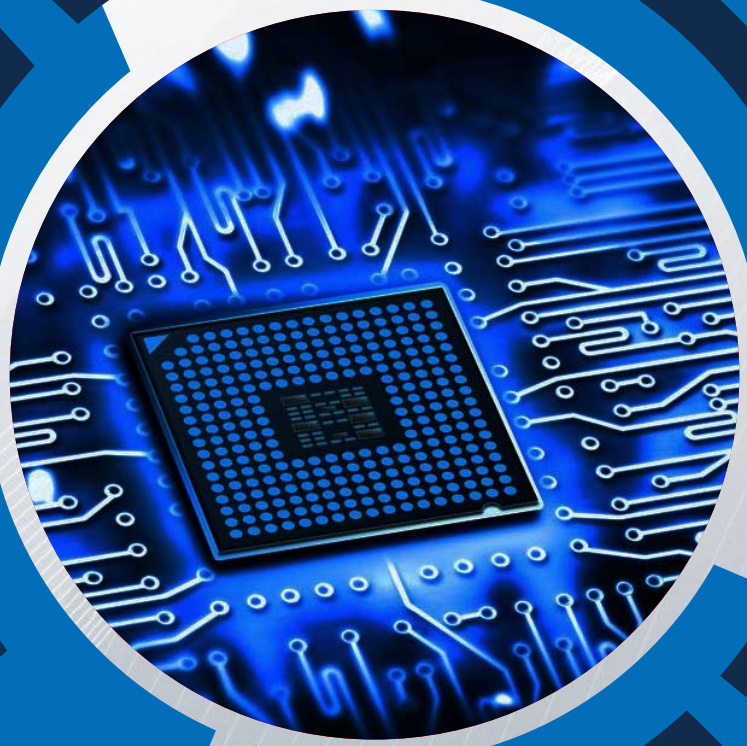
Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

Aplikasi Teknologi RFID pada IoT



YAYASAN PRIMA AGUS TEKNIK

Aplikasi Teknologi RFID pada IoT

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas (UNDIP) Semarang dan dari Universitas Kristen Satya Wacana Salatiga (UKSW) Diponegoro Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

YAYASAN PRIMA AGUS TEKNIK

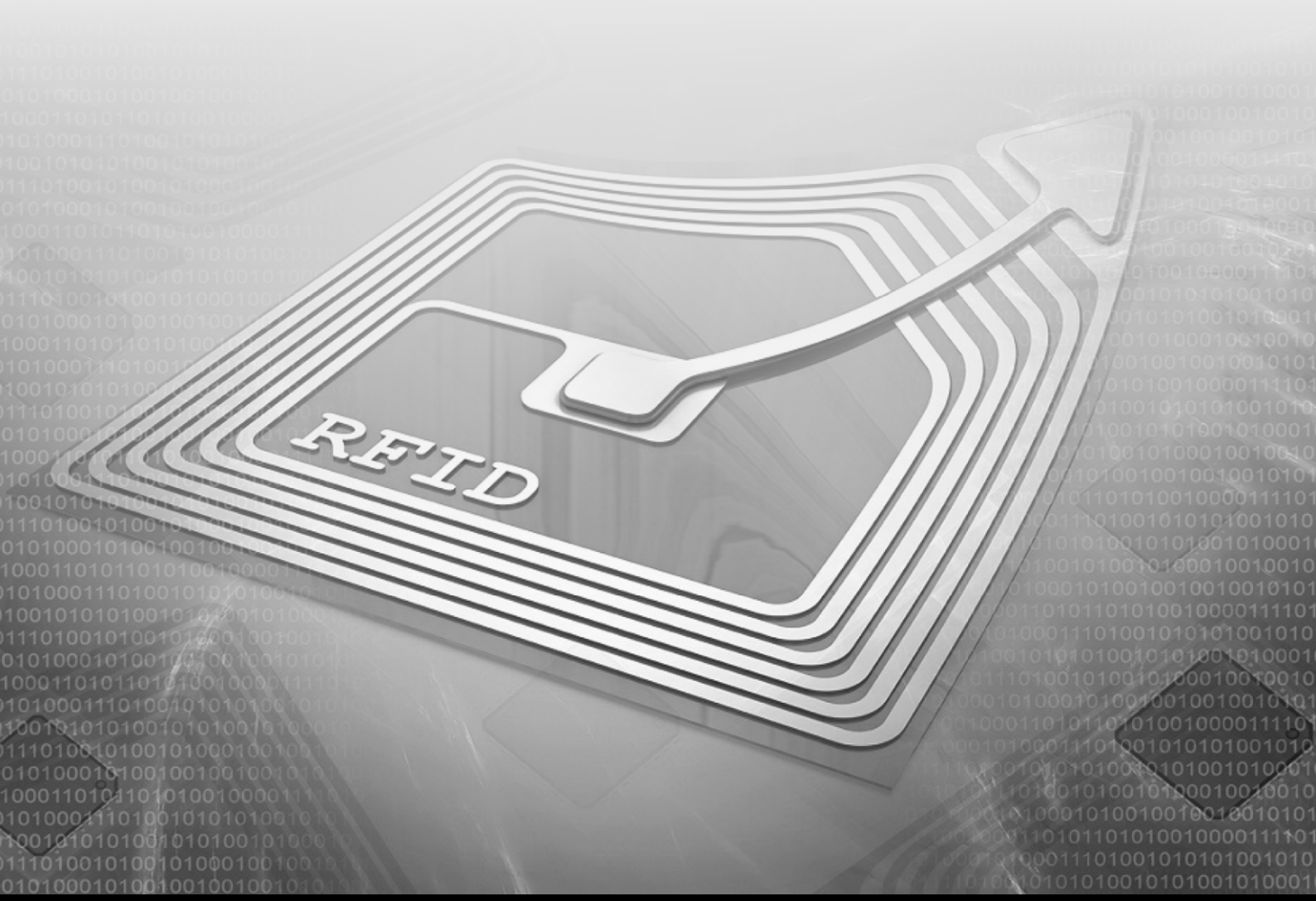
PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-20-0 (PDF)

9 786235 734200

Aplikasi Teknologi RFID pada IoT

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Aplikasi Teknologi RFID pada IoT

Penulis :
Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

ISBN : 9 786235 734200

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur pada Tuhan Yang Maha Esa bahwa buku yang berjudul "*Aplikasi Teknologi RFID pada IoT*" ini dapat diselesaikan dengan baik. Buku ini sangat bermanfaat bagi para pembacanya, karena buku ini menyajikan tentang teknologi RFID dalam penggunaanya di kehidupan sehari-hari. RFID merupakan suatu teknologi yang digunakan untuk melakukan pengambilan data dan identifikasi dengan menggunakan Barcode ataupun Magnetic Card. Dalam proses identifikasi, RFID membutuhkan dua perangkat, yaitu Tag dan Reader yang berfungsi dengan baik.

RFID Tag adalah alat yang menenmpel pada benda tertentu untuk selanjutnya akan di identifikasi oleh RFID reader. Tag RFID dibedakan menjadi 2 jenis Aktif dan Pasif perbedaan kedua Tag RFID ini hanya pada penggunaan catu daya (Baterai) yang selanjutnya akan dibahas dalam Bab 1 pada buku ini.

Bab 2 menyajikan protokol pencarian tag yang efisien berdasarkan vektor penyaringan dan mengevaluasi dampak kebisingan saluran pada kinerjanya. Kontribusi utama dalam bab ibu adalah metode iteratif ITSP berdasarkan Vektor Penyaringan sangatlah efektif dalam mengurangi jumlah data yang diperlukan anata tag dan pembaca, sehingga dapat menghemat waktu dalam proses pencarian / identifikasi. Dan pada Bab 3 disajikan pembahasan tentang memperkenalkan masalah otentikasi anonim dalam sistem RFID. Kami menyajikan protokol otentikasi anonim ringan menggunakan token yang dihasilkan secara dinamis. Protokol hanya membutuhkan komunikasi yang konstan dan overhead komputasi untuk pembaca dan tag.

Sedangkan pada Bab 4 atau akhir bab membahas masalah mengidentifikasi tag jaringan. Dua protokol identifikasi tag diberikan dan dibandingkan secara rinci. Temuan penting dalam bab ini salahsatunya adalah bahwa ketidakseimbangan beban menyebabkan biaya energi yang cukup besar pada tag, yang akan dijelaskan secara rinci dalam bab terakhir. Akhir kata semoga buku ini bermanfaat bagi para pembacanya.

Semarang, Desember 2021

Penulis

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 *INTERNET OF THINGS*

Internet of Things (IoT) adalah paradigma jaringan baru untuk sistem cyber-fisik yang memungkinkan objek fisik untuk mengumpulkan dan bertukar data. Di IoT, objek fisik dan agen siber dapat dirasakan dan dikendalikan dari jarak jauh di seluruh infrastruktur jaringan yang ada, yang memungkinkan integrasi antara dunia fisik dan sistem berbasis komputer dan karenanya memperluas Internet ke dunia nyata. IoT dapat menemukan banyak aplikasi di perumahan pintar, pemantauan lingkungan, sistem perawatan medis dan kesehatan, pertanian, transportasi, dll. Karena potensi aplikasinya yang signifikan, IoT telah menarik banyak perhatian baik dari penelitian akademis maupun pengembangan industri.

1.2 TEKNOLOGI RFID

Umumnya, setiap objek fisik di IoT perlu ditambah dengan beberapa teknologi ID otomatis sehingga objek dapat diidentifikasi secara unik. *Radio Frequency Identification* (RFID) adalah salah satu teknologi auto-ID yang paling banyak digunakan. Teknologi RFID mengintegrasikan komponen komunikasi, penyimpanan, dan komputasi sederhana dalam tag yang dapat dilampirkan yang dapat berkomunikasi dengan pembaca secara nirkabel melalui jarak jauh. Oleh karena itu, teknologi RFID menyediakan cara sederhana dan murah untuk menghubungkan objek fisik ke IoT—selama objek membawa tag, objek tersebut dapat diidentifikasi dan dilacak oleh pembaca.

Teknologi RFID telah banyak digunakan dalam berbagai aplikasi, seperti manajemen inventaris, rantai pasokan, pelacakan produk, transportasi, logistik, dan pengumpulan tol. Menurut riset pasar yang dilakukan oleh IDTechEx, ukuran pasar RFID telah mencapai Rp 133.350 pada tahun 2014, dan diproyeksikan meningkat menjadi Rp 409.650 setelah satu dekade.

Biasanya, sistem RFID terdiri dari sejumlah besar tag RFID, satu atau beberapa pembaca RFID, dan server backend. Tag komersial saat ini dapat diklasifikasikan ke dalam tiga kategori: (1) tag pasif, yang ditenagai oleh gelombang radio dari pembaca RFID dan berkomunikasi dengan pembaca melalui hamburan balik; (2) tag aktif, yang ditenagai oleh sumber energinya sendiri; dan (3) tag semi-aktif, yang menggunakan sumber energi internal untuk memberi daya pada sirkuitnya saat berkomunikasi dengan pembaca melalui hamburan balik. Sebagaimana ditentukan dalam protokol EPC Class-1 Gen-2 (C1G2), setiap tag memiliki ID unik yang mengidentifikasi objek yang dilampirkannya. Objek tersebut dapat berupa kendaraan, produk di gudang, e-paspor yang membawa informasi pribadi, perangkat medis yang merekam data kesehatan pasien, atau objek fisik lainnya di IoT. *Transceiver* terintegrasi dari setiap tag

memungkinkannya untuk mengirim dan menerima sinyal radio. Oleh karena itu, pembaca dapat berkomunikasi dengan tag dari jarak jauh selama tag tersebut berada di area interogasinya. Namun, komunikasi di antara tag RFID umumnya tidak dapat dilakukan karena daya transmisinya yang rendah. Tag jaringan yang muncul membawa peningkatan mendasar pada tag RFID dengan memungkinkan tag untuk berkomunikasi satu sama lain. Tag jaringan terintegrasi dengan komponen pemanen energi yang dapat memanen energi dari lingkungan sekitar.

Meluasnya penggunaan tag RFID di IoT membawa masalah baru pada efisiensi, keamanan, dan privasi yang sangat berbeda dari sistem jaringan tradisional. Buku ini menyajikan beberapa protokol RFID mutakhir yang bertujuan untuk meningkatkan efisiensi, keamanan, dan privasi IoT.

1.3 MASALAH PENCARIAN TAG

Diberikan satu set ID untuk tag yang diinginkan, masalah pencarian tag adalah mengidentifikasi tag yang diinginkan yang ada dalam sistem RFID. Perhatikan bahwa mungkin ada tag lain yang bukan milik set. Sebagai contoh, produsen menemukan bahwa beberapa produknya, yang telah didistribusikan ke gudang yang berbeda, mungkin rusak, dan ingin menariknya kembali untuk pemeriksaan lebih lanjut. Karena setiap produk di IoT membawa tag, dan produsen mengetahui semua ID tag dari produk yang rusak tersebut, ia dapat melakukan pencarian tag di setiap gudang untuk mengidentifikasi produk yang perlu ditarik kembali.

Untuk memenuhi persyaratan penundaan yang ketat dari aplikasi dunia nyata, efisiensi waktu adalah metrik kinerja penting untuk masalah pencarian tag RFID. Misalnya, sangat diinginkan untuk melakukan pencarian dengan cepat di gudang yang sibuk karena proses pencarian yang lama dapat mengganggu aktivitas lain yang memindahkan barang masuk dan keluar dari gudang. Satu-satunya pekerjaan sebelumnya yang mempelajari masalah ini disebut CATS yang tidak bekerja dengan baik di bawah beberapa kondisi umum (misalnya, jika ukuran set yang diinginkan jauh lebih besar daripada jumlah tag di area cakupan perangkat).

Kami menyajikan metode pencarian tag cepat berdasarkan teknik baru yang disebut vektor penyaringan. Vektor pemfilteran adalah larik bit satu dimensi ringkas yang dibuat dari ID tag, yang dapat digunakan untuk memfilter tag yang tidak diinginkan. Menggunakan vektor pemfilteran, kami merancang, menganalisis, dan mengevaluasi protokol pencarian tag berulang baru, yang secara progresif meningkatkan akurasi hasil pencarian dan mengurangi waktu setiap iterasi seminimal mungkin dengan menggunakan informasi yang dipelajari dari iterasi sebelumnya. Mengingat persyaratan akurasi, protokol iteratif akan berakhir setelah hasil pencarian memenuhi persyaratan akurasi. Kami menunjukkan bahwa protokol kami berkinerja jauh lebih baik daripada protokol CATS dan alternatif lain yang digunakan untuk perbandingan. Selain itu, protokol kami dapat diperluas untuk bekerja di bawah saluran yang bising dengan sedikit peningkatan waktu eksekusi.

1.4 OTENTIKASI RFID ANONIM

Proliferasi tag RFID dengan cara tradisional mereka membuat operator mereka dapat dilacak. Jika tag masa depan menembus ke dalam produk sehari-hari di IoT dan dibawa-bawa (seringkali tanpa disadari), privasi orang akan menjadi perhatian serius. Tag tipikal akan secara otomatis mengirimkan ID-nya sebagai tanggapan atas kueri dari pembaca terdekat. Jika kita membawa tag di saku kita atau di mobil kita, tag ini akan memberikan ID mereka kepada setiap pembaca yang menanyakannya, memungkinkan orang lain untuk melacak kita. Sebagai contoh, untuk seseorang yang mobilnya membawa tag (pembayaran tol otomatis atau pelat tag), ia mungkin tidak sadar dilacak selama bertahun-tahun oleh pintu tol atau orang lain yang memasang pembaca di lokasi yang menarik untuk belajar kapan dan di mana dia berada. Untuk melindungi privasi pembawa tag, kita perlu menemukan cara untuk menjaga kegunaan tag saat melakukannya secara anonim.

Banyak aplikasi RFID seperti pembayaran tol memerlukan otentikasi. Pembaca akan menerima informasi tag hanya setelah mengautentikasi tag dan sebaliknya. Otentikasi anonim harus melarang transmisi informasi pengenal apa pun, seperti ID tag, pengidentifikasi kunci, atau nomor tetap apa pun yang dapat digunakan untuk tujuan identifikasi. Akibatnya, muncul tantangan bagaimana pembaca yang sah dapat secara efisien mengidentifikasi kunci yang tepat untuk autentikasi tanpa informasi pengidentifikasi tag?

Pentingnya dan tantangan otentikasi anonim menarik banyak perhatian dari komunitas riset RFID. Banyak protokol otentikasi anonim telah dirancang. Namun, kami akan menunjukkan bahwa semua pekerjaan sebelumnya memiliki beberapa masalah potensial, baik menimbulkan komputasi tinggi atau overhead komunikasi, atau memiliki masalah keamanan atau fungsional. Selain itu, sebagian besar pekerjaan sebelumnya, jika tidak semuanya, menggunakan fungsi hash kriptografis, yang membutuhkan perangkat keras yang cukup besar, untuk mengacak data otentikasi agar tag tidak dapat dilacak. Persyaratan perangkat keras yang tinggi membuat mereka tidak cocok untuk tag berbiaya rendah dengan sumber daya perangkat keras yang terbatas. Oleh karena itu, merancang protokol otentikasi anonim untuk tag berbiaya rendah tetap menjadi masalah yang terbuka dan menantang.

Dalam desain kami, kami membuat perubahan mendasar dari paradigma tradisional untuk otentikasi RFID anonim. Pertama, kami melepaskan tag RFID yang dibatasi sumber daya dari penerapan fungsi rumit apa pun (mis., hash kriptografis). Karena pembaca tidak diperlukan dalam jumlah besar seperti halnya tag, mereka dapat memiliki lebih banyak sumber daya perangkat keras. Oleh karena itu, kami mengikuti prinsip desain asimetri untuk mendorong sebagian besar kerumitan kepada pembaca sambil membiarkan tag sesederhana mungkin. Kedua, kami mengembangkan teknik baru untuk menghasilkan token acak sesuai permintaan untuk otentikasi anonim. Protokol kami hanya memerlukan O.1/ *overhead* komunikasi dan *overhead* komputasi online per autentikasi untuk pembaca dan tag, yang merupakan

peningkatan yang signifikan dibandingkan teknologi sebelumnya. Oleh karena itu, protokol kami dapat diskalakan untuk sistem RFID besar. Akhirnya, analisis teoretis ekstensif, analisis keamanan, simulasi, dan uji keacakan statistik disediakan untuk memverifikasi efektivitas protokol kami.

1.5 IDENTIFIKASI TAG JARINGAN

Tag jaringan yang muncul menjanjikan untuk membawa peningkatan mendasar pada tag RFID tradisional dengan memungkinkan tag untuk berkomunikasi satu sama lain. Contoh dari tag tersebut adalah kelas baru dari tag jaringan pemanen energi ultra-rendah daya yang dirancang dan dibuat prototipe di Universitas Columbia. Objek yang ditandai yang secara tradisional tidak berjejaring di antara mereka sendiri, misalnya produk gudang, buku, furnitur, dan pakaian, kini dapat membentuk jaringan, yang memberikan fleksibilitas besar dalam aplikasi. Pertimbangkan gudang besar di mana sejumlah besar pembaca dan antena harus dikerahkan untuk menyediakan cakupan penuh. Penyebaran sistem seperti itu bisa sangat mahal. Selain itu, rintangan dan tumpukan objek yang ditandai dapat mencegah sinyal menembus ke setiap sudut penerapan, menyebabkan pembaca gagal mengakses beberapa tag. Masalah ini akan terpecahkan jika tag dapat menyampaikan transmisi ke pembaca yang tidak dapat diakses. Oleh karena itu, kami membayangkan bahwa tag jaringan akan memainkan peran penting dalam IoT sebagai peningkatan teknologi RFID saat ini.

Fitur baru dari tag jaringan membuka peluang penelitian baru. Kami fokus pada masalah identifikasi tag, yaitu mengumpulkan ID dari semua tag dalam suatu sistem. Ini adalah masalah yang paling mendasar untuk sistem RFID, tetapi belum dipelajari dalam konteks tag jaringan, di mana seseorang dapat memanfaatkan kemampuan jaringan untuk memfasilitasi proses pengumpulan ID, misalnya, mengumpulkan ID tag yang tidak berada dalam jangkauan pembaca.

Di luar jangkauan pembaca, tag jaringan ditenagai oleh baterai atau sumber energi yang dapat diisi ulang yang secara oportunistik memanen energi surya, piezoelektrik, atau panas dari lingkungan sekitar. Oleh karena itu, efisiensi energi adalah kriteria kinerja orde pertama untuk operasi yang dilakukan oleh tag jaringan. Selain itu, kita juga harus membuat proses identifikasi tag menjadi efisien waktu sehingga dapat menskalakan ke sistem tag yang besar di mana saluran komunikasi bekerja pada tingkat yang sangat rendah untuk konservasi energi.

Identifikasi RFID yang ada protokol tidak dapat diterapkan untuk mengidentifikasi tag jaringan karena mereka menganggap bahwa pembaca dapat menjangkau semua tag secara langsung. Kami membuat perubahan mendasar dalam desain protokol untuk tag jaringan, dan menghadirkan dua solusi: protokol pengumpulan ID berbasis pertentangan dan protokol pengumpulan ID serial. Kami mengungkapkan bahwa desain protokol berbasis pertentangan tradisional akan menimbulkan terlalu banyak overhead dalam sistem tag jaringan multihop karena semakin meningkatnya tabrakan dalam jaringan terhadap pembaca, yang menghasilkan

biaya energi yang berlebihan. Sebaliknya, desain terkoordinasi pembaca yang mencoba untuk membuat serial transmisi tag berkinerja jauh lebih baik. Selain itu, kami menunjukkan bahwa penyeimbangan beban sangat penting dalam mengendalikan biaya energi kasus terburuk yang ditimbulkan pada tag. Kami menemukan bahwa biaya energi kasus terburuk tinggi untuk desain protokol berbasis pertentangan dan serial karena beban yang tidak seimbang dalam jaringan. Namun, untuk protokol pengumpulan ID serial, kami dapat memberikan solusi berdasarkan nomor seri yang menyeimbangkan beban dan mengurangi biaya energi kasus terburuk.

BAB 2
EFISIENSI PENCARIAN TAG DALAM SISTEM RFID

Bab ini memperkenalkan masalah pencarian tag dalam sistem RFID besar. Sebuah teknik baru yang disebut vektor penyaringan dirancang untuk mengurangi transmisi overhead selama proses pencarian, sehingga meningkatkan efisiensi waktu. Berdasarkan teknik ini, kami menyajikan protokol pencarian tag berulang. Beberapa tag disaring di setiap putaran dan proses pencarian pada akhirnya akan berhenti ketika hasilnya memenuhi persyaratan akurasi yang diberikan. Selain itu, protokol diperluas untuk bekerja di bawah saluran yang bising. Hasil simulasi menunjukkan bahwa protokol kami berkinerja jauh lebih baik daripada pekerjaan terbaik yang ada.

Sisa bab ini disusun sebagai berikut. Bagian 2.1 memberikan model sistem dan pernyataan masalah. Bagian 2.2 secara singkat memperkenalkan pekerjaan terkait. Bagian 2.3 menjelaskan protokol baru kami secara rinci. Bagian 2.4 membahas saluran nirkabel yang bising. Bagian 2.5 mengevaluasi kinerja protokol kami dengan simulasi. Bagian 2.6 memberikan ringkasan.

2.1 MODEL SISTEM DAN PERNYATAAN MASALAH

2.1.1 Model Sistem

Kami mempertimbangkan sistem RFID yang terdiri dari satu atau lebih pembaca, server backend, dan sejumlah besar tag. Setiap tag memiliki ID 96-bit yang unik sesuai dengan standar EPC global Class-1 Gen-2 (C1G2). Sebuah tag dapat berkomunikasi dengan pembaca secara nirkabel dan melakukan beberapa perhitungan seperti hashing. Server backend bertanggung jawab untuk penyimpanan data, pemrosesan informasi, dan koordinasi. Ia mampu melakukan komputasi kinerja tinggi. Setiap pembaca terhubung ke server backend melalui tautan kabel atau nirkabel berkecepatan tinggi. Jika ada banyak pembaca (atau antena), kami membaginya menjadi grup yang tidak mengganggu dan protokol RFID apa pun dapat dilakukan untuk satu grup pada satu waktu, dengan pembaca dalam grup tersebut mengeksekusi protokol secara paralel. Pembaca dalam setiap kelompok dapat dikatakan sebagai satu kesatuan yang utuh, masih disebut pembaca karena kesederhanaannya. Banyak karya tentang koordinasi multi-pembaca dapat ditemukan dalam literatur.

Dalam praktiknya, laju transmisi tag-ke-pembaca dan laju transmisi pembaca-ke-tag mungkin berbeda dan tergantung pada lingkungan. Misalnya, seperti yang ditentukan dalam standar EPC global Kelas-1 Gen-2, kecepatan transmisi tag-ke-pembaca adalah 40–640kbps dalam format pengkodean FM0 atau 5-320kbps dalam format pengkodean subcarrier termodulasi Miller, sedangkan pembaca- tingkat transmisi to-tag adalah sekitar 26,7-128kbps. Namun, untuk menyederhanakan diskusi kami, kami menganggap tingkat transmisi tag-ke-

pembaca dan tingkat transmisi pembaca-ke-tag adalah sama, dan sangat mudah untuk mengadaptasi protokol kami untuk tingkat transmisi asimetris.

2.1.2 Slot Waktu

Pembaca RFID dan tag di area jangkauannya menggunakan protokol MAC slotted berbingkai untuk berkomunikasi. Kami berasumsi bahwa jam pembaca dan semua tag dalam sistem RFID disinkronkan oleh sinyal pembaca. Selama setiap frame, komunikasi diinisialisasi oleh pembaca dalam mode permintaan-dan-tanggapan, yaitu pembaca menyiarkan permintaan dengan beberapa parameter ke tag dan kemudian menunggu tag untuk membalas di slot waktu berikutnya.

Pertimbangkan slot waktu yang sewenang-wenang. Kami menyebutnya slot kosong jika tidak ada tag yang membalas di slot ini, atau slot sibuk jika satu atau lebih tag merespons di slot ini. Umumnya, tag hanya perlu mengirimkan informasi satu bit untuk membuat saluran sibuk sehingga pembaca dapat merasakan keberadaannya. Pembaca menggunakan "0" untuk mewakili slot kosong dengan saluran idle dan "1" untuk slot sibuk dengan saluran sibuk. Panjang slot tag untuk mengirimkan respons pendek satu bit dilambangkan sebagai ts. Perhatikan bahwa ts dapat diatur lebih besar dari waktu transmisi data satu bit untuk toleransi yang lebih baik terhadap penyimpangan jam dalam tag. Beberapa pekerjaan RFID sebelumnya membutuhkan jenis slot lain untuk transmisi ID tag, yang akan segera diperkenalkan.

2.1.3 Pernyataan Masalah

Misalkan kita tertarik pada kumpulan ID tag X x_1 yang diketahui; x_2; x_3; , setiap xi X disebut tag yang diinginkan. Misalnya, set mungkin berisi ID tag pada jenis produk tertentu yang sedang ditarik oleh produsen. Biarkan Y y_1; y_2; y_3; menjadi kumpulan tag dalam area cakupan sistem RFID (mis., Di gudang). Setiap x_i atau y_i mewakili ID tag. Masalah pencarian tag adalah untuk mengidentifikasi subset W dari tag yang diinginkan yang ada di area cakupan. Yaitu, $W \subseteq X$. Karena setiap tag di W berada di area cakupan, $W \subseteq Y$. Oleh karena itu, $W = X \cap Y$. Kami mendefinisikan rasio persimpangan X dan Y sebagai

$$R_{INTS} = \frac{|W|}{\min\{|X|, |Y|\}}. \qquad (2.1)$$

Menemukan W secara tepat bisa mahal jika X dan Y sangat besar. Jauh lebih efisien untuk menemukan W kira-kira, memungkinkan kesalahan terbatas kecil—semua tag yang diinginkan di area cakupan harus diidentifikasi, tetapi beberapa tag yang diinginkan yang tidak ada dalam cakupan mungkin secara tidak sengaja disertakan.[1]

Solusi kami bekerja secara iteratif. Setiap putaran mengesampingkan beberapa tag di X ketika menjadi pasti bahwa mereka tidak berada di area cakupan (yaitu, Y), dan juga mengesampingkan beberapa tag di Y ketika menjadi pasti bahwa mereka tidak diinginkan di X. Aturan ini tag -out disebut tag non-kandidat. Tag lain yang tetap mungkin berada di X dan Y disebut tag kandidat. Pada awalnya, hasil pencarian diinisialisasi ke semua tag X yang

diinginkan. Karena solusi kami dijalankan secara iteratif, hasil pencarian menyusut ke arah W ketika semakin banyak non-kandidat yang dikesampingkan.

Biarkan W* menjadi hasil pencarian akhir. Kami memiliki dua persyaratan berikut:

1. Semua tag yang diinginkan di area cakupan harus terdeteksi, yaitu W W*.
2. Positif palsu terjadi ketika tag di XW termasuk dalam W*, yaitu, tag yang tidak berada di area cakupan disimpan dalam hasil pencarian oleh pembaca.2 Rasio positif palsu adalah probabilitas untuk setiap tag di XW berada di W* setelah eksekusi protokol pencarian. Kami ingin mengikat rasio positif palsu dengan persyaratan sistem PREQ yang ditentukan sebelumnya, yang nilainya ditetapkan oleh pengguna. Dengan kata lain, kami mengharapkan

$$\frac{|W^* - W|}{|X - W|} \leq P_{REQ}. \qquad (2.2)$$

Notasi yang digunakan dalam bab ini diberikan pada Tabel 2.1 untuk referensi cepat.

2.2 PEKERJAAN TERKAIT

2.2.1 Identifikasi Tag

Solusi langsung untuk masalah pencarian tag adalah mengidentifikasi semua tag yang ada di Y. Setelah itu, kita dapat menerapkan operasi perpotongan X Y untuk menghitung W. Standar EPC C1G2 mengasumsikan bahwa pembaca hanya dapat membaca satu ID tag pada satu waktu. Dynamic Framed Slotted ALOHA (DFSA) diimplementasikan untuk menangani tabrakan tag, di mana setiap frame terdiri dari sejumlah slot dengan durasi yang sama.

Tabel 2.1 Notasi

Simbol	deskripsi
X	Set tag yang dicari
Y	Set tag dalam sistem RFID
W	Persimpangan X dan Y, yaitu W = X ∩ Y
X_i	Kumpulan tag kandidat yang tersisa di X, yaitu, hasil pencarian di awal putaran ke-i dari protokol kami;
Y_i	Kumpulan tag kandidat yang tersisa di Y di awal putaran ke-i dari protokol kami
u_i	Perbedaan antara X_i dan W, yaitu, $U_i = X_i$ — W
V_i	Perbedaan antara Y_i dan W, yaitu, $V_i = Y_i$ — W
$\lvert \cdot \rvert$	Kardinalitas himpunan
$h(\cdot)$	Fungsi hash yang seragam
$FV(\cdot)$	Memfilter vektor dari satu set

Terbukti bahwa batas atas teoritis throughput identifikasi menggunakan DFSA adalah sekitar $\frac{1}{e}$ tag per slot (e adalah konstanta alami), yang dicapai ketika ukuran bingkai diatur sama dengan jumlah tag yang tidak teridentifikasi. Sebagaimana ditentukan dalam EPC C1G2, setiap slot terdiri dari transmisi perintah QueryAdjust atau QueryRep dari pembaca, satu ID tag, dan dua nomor acak 16-bit: satu untuk reservasi saluran (penghindaran tabrakan) yang dikirim oleh tag, dan lainnya untuk ACK/NAK yang dikirimkan oleh reader. Kami menyatakan durasi setiap slot untuk identifikasi tag sebagai t_l. Oleh karena itu, batas bawah waktu identifikasi untuk tag di Y menggunakan DFSA adalah

$$T_{DFSA} = e \times |Y| \times t_l. \qquad (2.3)$$

Salah satu batasan DFSA saat ini adalah informasi yang terkandung dalam slot tabrakan terbuang sia-sia. Beberapa karya terbaru berfokus pada teknik Pemulihan Tabrakan (CR), yang memungkinkan resolusi beberapa ID tag dari slot tumbukan. Manfaat dari teknik CR, throughput identifikasi dapat ditingkatkan secara dramatis hingga 3,1 tag per slot. Misalkan throughput adalah v tag per slot setelah mengadopsi teknik CR. Batas bawah untuk waktu identifikasi adalah

$$T_{CR} = \frac{|Y|}{\upsilon} \times t_l. \qquad (2.4)$$

Perhatikan bahwa setelah menggunakan teknik CR, durasi sebenarnya dari setiap slot bisa lebih lama dari tl. Alasannya adalah bahwa pembaca mungkin perlu mengenali beberapa tag dan tag mungkin perlu mengirim pesan tambahan untuk memfasilitasi pemulihan tabrakan.

2.2.2 Protokol Polling

Protokol polling memberikan solusi alternatif untuk masalah pencarian tag. Alih-alih mengumpulkan semua ID di Y, pembaca dapat menyiarkan ID di X satu per satu. Setelah menerima ID, setiap tag memeriksa apakah ID yang diterima identik dengan miliknya sendiri. Jika demikian, tag mengirimkan respons singkat satu bit untuk memberi tahu pembaca tentang keberadaannya; jika tidak, tag tetap diam. Oleh karena itu, waktu eksekusi protokol polling adalah

$$T_{Polling} = |X| \times (t_{id} + t_s), \qquad (2.5)$$

dimana t_{id} adalah biaya waktu bagi pembaca untuk menyiarkan ID tag.
Protokol polling sangat efisien ketika X kecil. Namun, ia juga memiliki batasan serius. Pertama, itu tidak bekerja dengan baik ketika X Y . Kedua, konsumsi energi tag (terutama ketika tag aktif digunakan) signifikan karena tag di Y harus terus-menerus mendengarkan saluran dan menerima sejumlah besar ID hingga ID-nya sendiri diterima.

2.2.3 Protokol CATS

Untuk mengatasi masalah identifikasi tag dan protokol polling, Zheng et al. merancang protokol dua fase bernama Compact Approximator based Tag Searching protocol (CATS), yang merupakan solusi paling efisien untuk masalah pencarian tag hingga saat ini.

Ide utama dari protokol CATS adalah untuk mengkodekan ID tag ke filter Bloom dan kemudian mengirimkan filter Bloom bukan ID. Pada fase pertama, pembaca mengkodekan semua ID tag yang diinginkan di X ke dalam filter Bloom L_1-bit, dan kemudian menyiarkan filter ini bersama dengan beberapa parameter ke tag di area cakupan. Setelah menerima filter Bloom ini, setiap tag menguji apakah itu milik himpunan X. Jika jawabannya negatif, tag tersebut bukan kandidat dan akan diam selama sisa waktu. Setelah penyaringan fase satu, jumlah tag kandidat di Y berkurang. Selama fase kedua, tag kandidat yang tersisa di Y melaporkan keberadaan mereka dalam filter Bloom L_2-bit kedua yang dibuat dari kerangka slot waktu t_s. Setiap tag kandidat mentransmisikan dalam k slot yang dipetakan. Mendengarkan saluran, pembaca membangun filter Bloom berdasarkan status slot waktu: "0" untuk slot idle di mana tidak ada tag yang mentransmisikan, dan "1" untuk slot sibuk di mana setidaknya satu tag mentransmisikan. Dengan menggunakan filter Bloom ini, pembaca melakukan penyaringan untuk ID di X untuk melihat mana yang termasuk dalam Y, dan hasilnya dianggap sebagai $X \cap Y$.

Dengan persyaratan rasio positif palsu yang ditentukan sebelumnya PREQ, protokol CATS menggunakan pengaturan optimal berikut untuk L_1 dan L_2:

$$L_1 = |X| \log_\phi \left(-\frac{\alpha |X|}{\beta |Y| \ln P_{REQ}} \right), \tag{2.6}$$

$$L_2 = \frac{|X|}{\ln \phi} \left(\ln P_{REQ} - \frac{\alpha}{\beta} \right), \tag{2.7}$$

di mana ϕ adalah konstanta yang sama dengan 0,6185, α dan β masing-masing adalah konstanta yang berkaitan dengan laju transmisi pembaca-ke-tag dan laju transmisi tag-ke-pembaca. Dalam CATS, penulis mengasumsikan ts adalah waktu yang dibutuhkan untuk mengirimkan data satu bit, dan , yaitu, laju transmisi reader-to-tag dan laju transmisi tag-to-reader adalah identik. Oleh karena itu, total waktu pencarian protokol CATS adalah

$$\begin{aligned} T_{CATS} &= (L_1 + L_2) \times t_s \\ &= |X| \left(\log_\phi \left(\frac{-|X|}{|Y| \ln P_{REQ}} \right) + \frac{\ln P_{REQ} - 1}{\ln \phi} \right) \times t_s. \end{aligned} \tag{2.8}$$

2.3 PROTOKOL PENCARIAN TAG CEPAT BERDASARKAN VEKTOR PENYARINGAN

Bagian ini menyajikan Protokol Pencarian Tag Iteratif (ITSP) untuk memecahkan masalah pencarian tag dalam sistem RFID skala besar. Kami akan mengabaikan kesalahan saluran untuk saat ini dan menunda subjek ini.

2.3.1 Motivasi

Meskipun protokol CATS mengambil langkah maju yang signifikan dalam memecahkan masalah pencarian tag, protokol ini masih memiliki beberapa kelemahan penting. Pertama, saat mengoptimalkan ukuran filter Bloom L_1 dan L_2, CATS memperkirakan | X ∩ Y | hanya sebagai | X |. Perkiraan kasar ini dapat menyebabkan overhead yang cukup besar ketika | X ∩ Y | menyimpang secara signifikan dari | X |.

Kedua, diasumsikan bahwa | X | < | Y | dalam desain dan turunan rumusnya. Pada kenyataannya, jumlah tag yang diinginkan mungkin jauh lebih besar daripada jumlah di area jangkauan sistem RFID. Misalnya, mungkin ada sejumlah besar | X | dari produk yang ditandai yang sedang ditarik kembali, tetapi karena produk tersebut didistribusikan ke banyak gudang, jumlah | Y | dari label di gudang tertentu mungkin jauh lebih kecil daripada | X |. Meskipun CATS masih dapat bekerja dalam kondisi | X | >> | Y | , itu akan menjadi kurang efisien seperti yang akan ditunjukkan oleh simulasi kami.

Ketiga, kinerja CATS sensitif terhadap persyaratan rasio positif palsu P_{REQ}. Performanya menurun ketika nilai P_{REQ} sangat kecil. Sementara simulasi di [28] set P_{REQ} = 5%, nilainya mungkin harus jauh lebih kecil dalam beberapa kasus praktis. Sebagai contoh, misalkan | X | = 100.000, dan | W | = 1000. Jika kita set P_{REQ} = 5%, jumlah tag buronan yang diklaim palsu di Y oleh CATS akan sampai | X – W | = P_{REQ} 4995, jauh lebih banyak dari 1000 tag buronan yang sebenarnya ada di Y.

Kami akan menunjukkan bahwa cara berulang dalam mengimplementasikan filter Bloom jauh lebih efisien daripada cara klasik yang diadopsi oleh protokol CATS.

2.3.2 Filter Bloom

Filter Bloom adalah struktur data kompak yang mengkodekan keanggotaan untuk satu set item. Untuk mewakili satu set S = { e_1; e_2; … ; e_n} menggunakan filter Bloom, kita membutuhkan array bit dengan panjang l di mana semua bit diinisialisasi ke nol. Untuk mengkodekan setiap elemen e ∉ S, kami menggunakan k fungsi hash, h_1, h_2, … , h_k, untuk memetakan elemen secara acak ke k bit dalam array bit, dan mengatur bit tersebut menjadi satu. Untuk pencarian keanggotaan elemen b, kita kembali memetakan elemen ke k bit dalam array dan melihat apakah semuanya adalah satu. Jika demikian, kami mengklaim bahwa b milik S; jika tidak, harus benar bahwa b ∉ S. Filter Bloom dapat menyebabkan false positive: elemen non-anggota diklaim secara salah sebagai anggota S. Probabilitas terjadinya false positive dalam pencarian keanggotaan diberikan sebagai berikut:

$$P_B = \left(1 - \left(1 - \frac{1}{l}\right)^{kn}\right)^k \approx \left(1 - e^{-kn/l}\right)^k. \qquad (2.9)$$

Ketika k D ln ~ l , P kira - kira diminimalkan menjadi $(\frac{1}{2})^k = (\frac{1}{2})^{In2\frac{1}{2}}$. Untuk mencapai nilai target P_B, ukuran minimum filter adalah $— \frac{InP_B}{(In\ 2)^2}\ n$. CATS mengirimkan satu filter Bloom

dari pembaca ke tag dan filter Bloom lainnya dari tag kembali ke pembaca. Perhatikan filter Bloom pertama yang mengkodekan X. Sebagai n = |X|, ukuran filternya adalah $-\frac{InP_B}{(In\,2)^2}\ n\ |X|$. Sebagai contoh, untuk mencapai P_B = 0.001, ukurannya menjadi 14.4 x |X| bit. Demikian pula, ukuran filter kedua dari tag ke pembaca juga terkait dengan probabilitas positif palsu target. Di bawah ini kami menunjukkan bahwa ukuran keseluruhan filter Bloom dapat dikurangi secara signifikan dengan merekonstruksinya sebagai vektor penyaringan dan kemudian menerapkan vektor-vektor ini secara iteratif.

2.3.3 Memfilter Vektor

Filter Bloom juga dapat diimplementasikan dengan cara yang tersegmentasi. Kami membagi array bitnya menjadi k segmen yang sama, dan fungsi hash ke-i akan memetakan setiap elemen ke bit acak di segmen ke-i, untuk i ∈ [1…k]. Kami menamai setiap segmen sebagai vektor pemfilteran (FV), yang memiliki l/k bit. Rumus berikut memberikan probabilitas positif palsu dari satu vektor pemfilteran, yaitu, probabilitas non-anggota untuk di-hash ke bit "1" dalam vektor:

$$P_{FV} = 1 - \left(1 - \frac{1}{l/k}\right)^n \approx 1 - e^{-kn/l}. \qquad (2.10)$$

Karena ada k segmen independen, probabilitas positif palsu keseluruhan dari filter Bloom tersegmentasi adalah

$$P_{FP} = (P_{FV})^k \approx \left(1 - e^{-kn/l}\right)^k, \qquad (2.11)$$

yang kira-kira sama dengan hasil pada (2.9). Artinya, kedua cara penerapan filter Bloom memiliki kinerja yang sama. Nilai PFP juga diminimalkan ketika k = ln 2 x $\frac{1}{n}$. Oleh karena itu, ukuran optimal dari setiap vektor penyaringan adalah

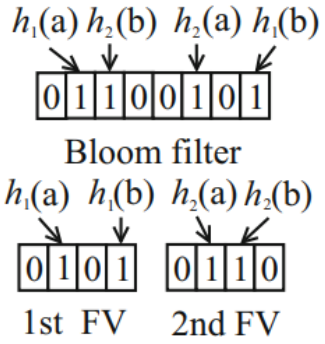
$$\frac{l}{k} = \frac{n}{\ln 2}, \qquad (2.12)$$

yang mengakibatkan

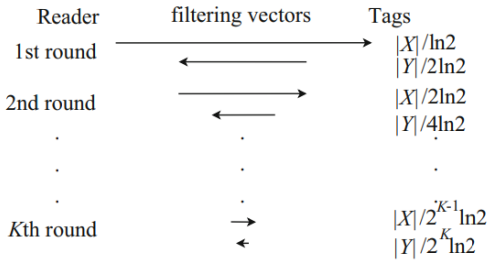
$$P_{FV} \approx \frac{1}{2}. \qquad (2.13)$$

Yaitu, setiap vektor pemfilteran rata-rata menyaring setengah dari non-anggota. Gambar 2.1 mengilustrasikan konsep penyaringan vektor. Misalkan kita memiliki dua elemen a dan b, dua fungsi hash h_1 dan h_2, dan array bit 8-bit. Pertama, misalkan h_1 (a) mod 8 = 1, h_1 (b) mod 8 = 7, h_2(a)mod 8 = 5, h_2(b) mod 8 = 2, dan kita membangun filter Bloom untuk a dan b di bagian atas gambar. Selanjutnya, kita membagi array bit menjadi dua vektor penyaringan 4-bit, dan menerapkan h1 ke segmen pertama dan h_2 ke segmen kedua. Karena h_1 (a) mod 4= 1, h_1(b) mod

4 = 3, h_2(a) mod 4 = 1, h_2(b) mod 4 = 2, kita membangun dua vektor penyaringan di bagian bawah gambar.



Gambar 2.1 Bloom filter dan filter filter



Gambar 2.2 Penggunaan berulang dari vektor penyaringan.

Setiap panah mewakili satu vektor pemfilteran, dan panjang panah menunjukkan ukuran vektor pemfilteran, yang ditentukan di sebelah kanan. Saat ukurannya menyusut di putaran berikutnya, jumlah total data yang dipertukarkan antara pembaca dan tag berkurang secara signifikan

2.3.4 Penggunaan Vektor Penyaringan Secara Iteratif

Dalam buku ini, kami menggunakan vektor pemfilteran dengan cara berulang yang baru: Filter Bloom antara pembaca dan tag dipertukarkan dalam putaran; satu vektor penyaringan dipertukarkan di setiap putaran, dan ukuran vektor penyaringan terus-menerus dikurangi di putaran berikutnya, sehingga ukuran keseluruhan setiap filter Bloom jauh berkurang.

Di bawah ini kami menggunakan contoh sederhana untuk menjelaskan ide tersebut, yang diilustrasikan pada Gambar 2.2: Misalkan tidak ada tag yang diinginkan di area jangkauan pembaca RFID, yaitu X ∩ Y = . Pada putaran pertama, pertama-tama kita mengkodekan X dalam vektor pemfilteran berukuran |X| = / ln 2 melalui fungsi hash h_1, dan menyiarkan vektor tersebut ke tag filter di Y. Dengan menggunakan fungsi hash yang sama, setiap tag kandidat di Y mengetahui bit mana di dalam vektor itu dipetakan, dan itu hanya perlu memeriksa nilai bit

itu. Jika bitnya nol, tag menjadi non-calon dan tidak akan berpartisipasi dalam eksekusi protokol lebih lanjut. Vektor penyaringan mengurangi jumlah tag kandidat di Y menjadi sekitar |Y| x P_{FV} ≈ |Y| /2. Kemudian vektor penyaringan dengan ukuran |Y| / (2 ln 2) dikirim dari tag kandidat yang tersisa di Y kembali ke pembaca dengan cara yang mirip dengan [28]: Setiap tag kandidat mem-hash ID-nya ke slot dalam kerangka waktu dan mengirimkan respon satu-bit di slot itu. Dengan mendengarkan status slot dalam kerangka waktu, pembaca menyusun vektor penyaringan, "1" untuk slot sibuk dan "0" untuk slot kosong. Pembaca menggunakan vektor ini untuk menyaring tag non-kandidat dari X. Setelah penyaringan, jumlah tag kandidat yang tersisa di X dikurangi menjadi sekitar |X| P_{FV} ≈ |X| / 2. Hanya tag kandidat di X yang perlu dikodekan dalam vektor penyaringan berikutnya, menggunakan fungsi hash yang berbeda h_2. Oleh karena itu, pada putaran kedua, ukuran vektor pemfilteran dari pembaca ke tag dikurangi setengahnya menjadi |X| / (2 ln 2), dan dengan cara yang sama ukuran vektor pemfilteran dari tag ke pembaca juga dikurangi setengahnya menjadi |Y| / (4 ln 2). Mengulangi proses di atas, mudah untuk melihat bahwa pada putaran ke-i, ukuran vektor pemfilteran dari pembaca ke tag adalah |X| / 2^{i-1} ln 2), dan ukuran vektor pemfilteran dari tag ke pembaca adalah |Y| / (2^i ln 2). Setelah K putaran, ukuran total semua vektor pemfilteran dari pembaca ke tag adalah

$$\frac{1}{\ln 2}\sum_{i=1}^{K}\frac{|X|}{2^{i-1}} < \frac{2|X|}{\ln 2}, \qquad (2.14)$$

di mana $\frac{2\,|X|}{In\,2}$ adalah batas atas, terlepas dari jumlah K putaran (yaitu, terlepas dari persyaratan pada probabilitas positif palsu). Ini lebih baik dibandingkan dengan CATS yang ukuran filternya, $-\frac{In\,P_B}{(In2)^2}\,|X|$, tumbuh terbalik di P_B, dan mencapai 14:4 $x\ |X|$ bit ketika P_B 0:001 dalam contoh kita sebelumnya.

Demikian pula, ukuran total semua vektor pemfilteran dari tag ke pembaca adalah

$$\frac{1}{\ln 2}\sum_{i=1}^{K}\frac{|Y|}{2^{i}} < \frac{|Y|}{\ln 2}, \qquad (2.15)$$

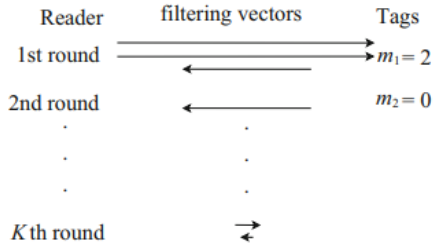
dan P_{FP} $(P_{FV})^K \approx (\frac{1}{2})^K$. Kita dapat membuat P_{FP} sekecil yang kita inginkan dengan meningkatkan n, sedangkan total overhead transmisi tidak pernah melebihi $\frac{1}{In\,2}\ (2|X| + |Y|)$ bit. Kekuatan vektor pemfilteran dalam filtrasi dua arah terletak pada kemampuannya untuk mengurangi kumpulan kandidat selama setiap putaran, sehingga mengurangi ukuran vektor pemfilteran pada putaran berikutnya dan dengan demikian menghemat waktu. Kekuatannya untuk mengurangi vektor penyaringan berikutnya terkait dengan |X - W| dan |Y - W| . Semakin banyak jumlah tag di luar W, semakin banyak mereka akan disaring di setiap putaran, dan semakin besar efek pengurangannya.

2.3.5 Pendekatan Umum

Tidak seperti protokol CATS, pendekatan berulang kami membagi penyaringan dua arah dalam proses pencarian tag menjadi beberapa putaran. Sebelum putaran ke-i, himpunan kandidat tag di X dilambangkan sebagai X_i ($\subseteq$ X), yang juga disebut hasil pencarian setelah putaran ke (i -1). Hasil pencarian akhir adalah kumpulan tag kandidat yang tersisa di X setelah semua putaran selesai. Sebelum putaran ke-i, himpunan kandidat tag di Y dilambangkan sebagai Y_i ($\subseteq$ Y). Awalnya, X_1 = X dan Y_1 = Y. Kami mendefinisikan $U_i = X_i - W$ dan $V_i = Y_i - W$, yang merupakan tag yang akan disaring. Karena W selalu merupakan subset dari X_i dan Y_i, kita memiliki

$$|U_i| = |X_i| - |W|$$
$$|V_i| = |Y_i| - |W|. \qquad (2.16)$$

Alih-alih menukar satu vektor pemfilteran pada satu waktu, kami menggeneralisasi pendekatan iteratif kami dengan mengizinkan beberapa vektor pemfilteran dikirim secara berurutan. Setiap putaran terdiri dari dua fase. Pada fase satu putaran ke-i, pembaca RFID menyiarkan sejumlah mil vektor penyaringan, yang mengecilkan kumpulan tag kandidat yang tersisa di Y dari Y_i ke Y_{i+1}. Di fase dua putaran ke-i, satu penyaringan



Gambar 2.3 Pendekatan Umum.

Setiap putaran memiliki dua fase. Pada fase satu, pembaca mentransmisikan nol, satu, atau beberapa vektor penyaringan. Pada fase dua, tag mengirim tepat satu vektor pemfilteran ke pembaca. Pada contoh yang ditunjukkan oleh gambar, m_1 = 2 dan m_2 = 0, yang berarti ada dua vektor pemfilteran yang dikirim oleh pembaca pada putaran pertama, sedangkan tidak ada vektor pemfilteran dari pembaca selama putaran kedua, vektor dikirim dari tag kandidat yang tersisa di Y_{i+1} kembali ke pembaca, yang menggunakan vektor penyaringan yang diterima untuk mengecilkan kumpulan kandidat yang tersisa dari X_i ke X_{i+1}, mengatur panggung untuk babak berikutnya. Proses ini berlanjut sampai rasio positif palsu memenuhi persyaratan P_{REQ}.

Nilai m_i akan ditentukan pada subbagian berikutnya. Jika $m_i > 0$, beberapa vektor pemfilteran akan dikirim secara berurutan dari pembaca ke tag dalam satu putaran. Jika mi 0, tidak ada vektor penyaringan yang dikirim dari pembaca pada putaran ini. Ketika ini terjadi, pada dasarnya memungkinkan beberapa vektor pemfilteran dikirim secara berurutan dari tag ke pembaca (melalui beberapa putaran). Sebuah ilustrasi diberikan pada Gambar. 2.3.

2.3.6 Nilai m_i

Misalkan K adalah jumlah total putaran. Setelah semua putaran K, kami menggunakan X_{K+1} sebagai hasil pencarian kami. Ada total K vektor pemfilteran yang dikirim dari tag ke pembaca. Kita tahu dari bab 2.3.3 bahwa setiap vektor pemfilteran dapat menyaring setengah dari non-anggota (dalam kasus kami, tag dalam X - W). Untuk memenuhi persyaratan rasio positif palsu P_{REQ}, batasan berikut harus berlaku:

$$(P_{FV})^K \approx \left(\frac{1}{2}\right)^K \leq P_{REQ}. \qquad (2.17)$$

Oleh karena itu, nilai K diatur ke $[-\frac{InP_{REQ}}{In\,2}]$. (Kita akan membahas bagaimana menjamin pemenuhan persyaratan PREQ.) Selanjutnya, kita membahas bagaimana mengatur nilai mi, $1 \leq i \leq K$, untuk meminimalkan waktu eksekusi setiap putaran. Kami menggunakan FV (·)untuk menunjukkan vektor penyaringan dari suatu himpunan. Pada fase satu putaran ke-i, pembaca membuat vektor pemfilteran mi, dilambangkan sebagai FV_{i1} (X_i), FV_{i2} (X_i), … , FV_{imi} (X_i), yang disiarkan secara berurutan ke tag. Kita mengetahui ukuran setiap vektor penyaringan adalah $|X_i| / \ln 2$. Setelah penyaringan berdasarkan vektor-vektor ini, jumlah tag kandidat yang tersisa di Y_{i+1} rata-rata

$$\begin{aligned} |Y_{i+1}| &\approx |V_i| \times (P_{FV})^{m_i} + |W| \\ &\approx |V_i| \times (1/2)^{m_i} + |W| \\ &= |V_i|/2^{m_i} + |W|. \end{aligned} \qquad (2.18)$$

Pada fase dua putaran ke-i, tag di Y_{i+1} menggunakan kerangka waktu $\frac{1}{In\,2}$ x | Y_{i+1} | slot untuk melaporkan keberadaannya. Setelah menerima tanggapan, pembaca membangun vektor penyaringan, dilambangkan sebagai FV_i (Y_{i+1}). Setelah penyaringan berdasarkan $FV_i(Y_{i+1})$, ukuran hasil pencarian X_{i+1} rata-rata

$$\begin{aligned} |X_{i+1}| &\approx |U_i| \times P_{FV} + |W| \\ &\approx |U_i|/2 + |W| \\ &= (|X_i| + |W|)/2. \end{aligned} \qquad (2.19)$$

Kami menyatakan waktu transmisi putaran ke-i dengan f .mi/. Untuk membuat perbandingan yang adil dengan CATS, kami menggunakan pengaturan parameter yang sesuai dengan. Oleh karena itu $f(m_i) = \frac{1}{In\,2}$ x m_i x $|X_i|$ x t_s + $\frac{1}{In\,2}$ x | Y_{i+1} | x t_s , yang diset menjadi:

$$f(m_i) = \frac{t_s}{\ln 2}\,(m_i|X_i| + |V_i|/2^{m_i} + |W|)\,. \qquad (2.20)$$

Untuk mencari nilai mi yang meminimalkan $f(m_i)$, kita ambil turunan orde pertama dan atur ruas kanan ke nol.

$$\frac{df(m_i)}{dm_i} = \frac{t_s}{\ln 2}\,(|X_i| - \ln 2|V_i|/2^{m_i}) = 0 \qquad (2.21)$$

Oleh karena itu, nilai $f(m_i)$diminimalkan ketika

$$m_i = \frac{\ln(\ln 2|V_i|/|X_i|)}{\ln 2}. \qquad (2.22)$$

Karena mi tidak bisa menjadi bilangan negatif, kita reset $m_i = 0$ jika $\frac{In(In\ 2|V_i|/|X_i|)}{In\ 2} < 0$ Selanjutnya, mi harus berupa bilangan bulat. Jika $\frac{In(In\ 2|V_i|/|X_i|)}{In\ 2}$ bukan bilangan bulat, kita membulatkan mi ke langit-langit atau ke lantai, tergantung mana yang menghasilkan nilai $f(m_i)$ yang lebih kecil.

Untuk saat ini, kita asumsikan bahwa kita mengetahui |W| dan |Y| dalam perhitungan m_i. Nanti kami akan menunjukkan bagaimana memperkirakan nilai-nilai ini dengan cepat dalam pelaksanaan setiap putaran protokol kami. Awalnya, $|X_1|$ (= |X|) diketahui. $|V_1|$ dapat dihitung dari (2.16). Oleh karena itu, nilai m1 dapat dihitung dari (2.22). Setelah itu, kita dapat mengestimasi $|Y_2|$, $|X_2|$, dan $|V_2|$ berdasarkan (2.18), (2.19), dan (2.16), masing-masing. Dari $|X_2|$, dan $|V_2|$, kita dapat menghitung nilai m_2. Mengikuti prosedur yang sama, kita dapat menghitung secara iteratif semua nilai mi untuk $1 \leq I \leq K$.

Kami menemukan sering terjadi bahwa urutan mi memiliki beberapa nol berturut-turut di akhir, yaitu, $\exists p < K$, $m_i = 0$ untuk $I \in [p , K]$. Dalam hal ini, kami mungkin dapat lebih mengoptimalkan nilai m_p dengan sedikit penyesuaian. Kami pertama-tama menjelaskan alasan untuk $m_p = 0$: Membutuhkan waktu bagi pembaca untuk menyiarkan vektor pemfilteran di fase satu putaran pth. Memang benar bahwa vektor penyaringan ini dapat mengurangi set Y_p, sehingga mengurangi ukuran bingkai fase dua di putaran ke p. Namun, jika biaya waktu pengiriman vektor penyaringan tidak dapat dikompensasikan dengan pengurangan waktu fase dua, akan lebih baik untuk menghapus vektor penyaringan ini dengan mengatur $m_p = 0$. (Situasi ini biasanya terjadi di dekat akhir urutan mi karena jumlah tag yang tidak diinginkan dalam set kandidat yang tersisa Y_p sudah sangat kecil.) Tetapi jika semua nilai mi pada putaran berikutnya (setelah m_p) adalah nol, meningkatkan mp ke nilai non-nol $m_p = 0$ dapat membantu mengurangi waktu transmisi fase dua dari semua selanjutnya mengirimkan vektor penyaringan m_p itu.

Pertimbangkan waktu transmisi putaran (K - p + 1) ini secara keseluruhan, dilambangkan dengan $G(m'_p , p)$. Sangat mudah untuk diturunkan

$$G(m'_p, p) = \left(\frac{m'_p}{\ln 2}|X_p| + \frac{K-p+1}{\ln 2}\left(\frac{|V_p|}{2^{m'_p}} + |W|\right)\right) t_s. \quad (2.23)$$

Untuk meminimalkan G(m$^{l}_{p}$, p), kita punya

$$m'_p = \begin{cases} 0 & \text{if } \gamma < 0 \\ \gamma & \text{if } \gamma \geq 0 \end{cases} \quad (2.24)$$

dimana $= \frac{In\ (In\ 2\ (K-p+1)|V_p|/\ |X_p|}{In\ 2}$. Akibatnya, mp diperbarui ke m_p , sementara mi lainnya, dalam m$^{l}_{p}$, tetap tidak berubah. Di sini, kami memberikan contoh untuk mengilustrasikan cara menghitung nilai mi. Memperkirakan |X| = 5000, |Y| = 50,000, |W| = 500, dan P_{REQ} = 0:001, jadi $K\left[\frac{-In\ 0.001}{In\ 2}\right] = 10$. Dengan menggunakan (2.22), kita dapat menghitung nilai dari m_1 hingga m_{10}. Hasilnya tercantum pada Tabel 2.2. Ada urutan nol dari m_7 sampai m_{10}. Dengan demikian, kita dapat melakukan perbaikan menggunakan (2.24), dan hasil yang dioptimalkan ditunjukkan pada Tabel 2.3.

Tabel 2.2 Nilai awal m_i.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
3	1	0	1	0	1	0	0	0	0

Tabel 2.3 Nilai yang dioptimalkan dari mi.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
3	1	0	1	0	1	2	0	0	0

2.3.7 Protokol Pencarian Tag Iteratif

Setelah menghitung nilai mi, kami dapat menyajikan protokol pencarian tag berulang (ITSP) kami berdasarkan pendekatan umum di bab 2.3.5. Protokol terdiri dari K putaran iteratif. Setiap putaran terdiri dari dua fase. Pertimbangkan putaran ke-i, di mana $1 \leq i \leq K$.

2.3.7.1 Fase Satu

Pembaca RFID membuat vektor penyaringan mi untuk X_i menggunakan fungsi hash m_i. Menurut (2.12), kami menetapkan ukuran L_{Xi} dari setiap vektor pemfilteran sebagai

$$L_{X_i} = \frac{1}{\ln 2} \times |X_i|. \quad (2.25)$$

Pembaca RFID kemudian menyiarkan vektor penyaringan tersebut satu per satu. Setelah menerima vektor pemfilteran, setiap tag di Y_i memetakan ID-nya ke bit dalam vektor pemfilteran menggunakan fungsi hash yang sama yang digunakan pembaca untuk membuat filter. Tag memeriksa apakah bit ini "1". Jika demikian, itu tetap menjadi

tag kandidat; jika tidak, itu dikecualikan sebagai tag non-kandidat dan segera keluar dari proses pencarian. Kumpulan tag kandidat yang tersisa adalah Y_{i+1}.

Jika vektor penyaringan terlalu panjang, pembaca membagi setiap vektor menjadi blok dengan panjang tertentu (misalnya, 96 bit) dan mentransmisikan satu demi satu blok. Mengetahui bit mana yang dipetakan, setiap tag hanya perlu merekam satu blok yang berisi bitnya.

Dari (2.13), kita tahu bahwa probabilitas positif palsu setelah menggunakan vektor penyaringan mi adalah $(P_{FV})^{mi} \approx (1/2)^{mi}$. Oleh karena itu, $|Y_{i+1}| \approx |V_i| \times (P_{FV})^{mi} + |W| \approx |V_i| / 2^{mi} + |W|$

2.3.7.2 Fase Dua

Pembaca menyiarkan ukuran bingkai L_{Yi+1} fase dua ke tag, di mana

$$L_{Y_{i+1}} = \frac{1}{\ln 2}\left(|V_i|/2^{m_i} + |W|\right). \qquad (2.26)$$

Setelah menerima L_{Yi+1} , setiap tag di Y_{i+1} secara acak memetakan ID-nya ke slot dalam kerangka waktu menggunakan fungsi hash dan mengirimkan respons pendek satu bit ke pembaca di slot itu. Berdasarkan keadaan slot yang diamati (sibuk atau kosong) dalam kerangka waktu, pembaca membuat vektor pemfilteran, yang digunakan untuk memfilter non-kandidat dari Xi. Waktu transmisi keseluruhan dari semua putaran K di ITSP adalah

$$T_{ITSP} = \sum_{i=i}^{K}(m_i \times L_{X_i} + L_{Y_{i+1}}) \times t_s. \qquad (2.27)$$

2.3.8 Estimasi Kardinalitas

Ingat dari bab 2.3.6 bahwa kita harus mengetahui nilai $|X_i|$, $|W|$, dan $|V_i|$ untuk menentukan m_i, L_{Xi} , dan L_{Yi+1} . Sangat mudah untuk menemukan nilai $|X_i|$ dengan menghitung jumlah tag dalam hasil pencarian putaran ke (i — 1). Sementara itu, kita tahu $|V_i| \approx |V_{i-1}| / 2^{mi-1}$ dan $|V_1| = |Y_1| — |W|$. Oleh karena itu, kita hanya perlu memperkirakan $|W|$ dan $|Y_1|$

Selain berfungsi sebagai filter, vektor filter juga dapat digunakan untuk estimasi kardinalitas, fitur yang tidak dieksploitasi pada [28]. Karena tidak ada vektor penyaringan yang tersedia di awal, putaran pertama ITSP harus diperlakukan secara terpisah: Kami dapat menggunakan protokol estimasi kardinalitas yang efisien ART [26] untuk memperkirakan $|Y|$ (yaitu, $|Y_1|$) jika nilainya tidak diketahui pada awalnya . Adapun $|W|$, awalnya diasumsikan min ($|X|$, $|Y|$) .

Selanjutnya, kita dapat memanfaatkan vektor penyaringan yang diterima oleh pembaca pada fase dua putaran ke-i ($i \geq 1$) untuk memperkirakan $|W|$ tanpa pengeluaran transmisi tambahan. Proses estimasi adalah sebagai berikut: Pertama, menghitung jumlah sebenarnya

dari bit "1" dalam vektor penyaringan, dilambangkan sebagai N_1^*, kita tahu probabilitas positif palsu yang sebenarnya dari menggunakan vektor penyaringan ini, dilambangkan dengan P^*_i , adalah

$$P_i^* = N_1^*/L_{Y_{i+1}}, \qquad (2.28)$$

karena tag yang tidak diinginkan sewenang-wenang memiliki peluang N_1^* dari L_{Yi+1} untuk dipetakan ke bit "1", di mana L_{Yi+1} adalah ukuran vektor. Sementara itu, kita dapat mencatat jumlah tag dalam hasil pencarian sebelum dan sesudah putaran ke-i, yaitu $|X_i|$ dan $|X_{i+1}|$, kami memiliki $|X_i| = |U_i| + |W|$, $|X_{i+1}| = |U_{i+1}| + |W|$, dan $|U_{i+1}| \approx |U_i| \times P^*_i$ Karena itu,

$$|W| \approx \frac{|X_{i+1}| - |X_i| \times P_i^*}{1 - P_i^*}. \qquad (2.29)$$

Untuk tujuan akurasi, kita dapat memperkirakan |W| setelah setiap putaran, dan memperoleh nilai rata-rata.

2.3.9 Vektor Penyaringan Tambahan

Estimasi mungkin memiliki kesalahan. Menggunakan nilai m_i dan L_{Yi} yang dihitung dari estimasi |W| dan $|Y_i|$, konsekuensi langsungnya adalah rasio positif palsu yang sebenarnya, dilambangkan sebagai P_T, dapat lebih besar dari persyaratan P_{REQ}. Untungnya, dari (2.28), pembaca dapat menghitung rasio positif palsu aktual P^*_i, $1 \leq i \leq k$, dari setiap vektor penyaringan yang diterima dalam fase dua ITSP. Dengan demikian, kita memiliki

$$P_T = \prod_1^K P_i^*. \qquad (2.30)$$

Jika $P_T > P_{REQ}$, protokol kami akan secara otomatis menambahkan vektor penyaringan tambahan untuk menyaring X_{K+1} lebih lanjut hingga $P_T \geq P_{REQ}$ (seperti yang dijelaskan dalam Bagian 2.3.4).

2.3.10 Kebutuhan Perangkat Keras

ITSP tidak dapat didukung oleh tag off-the-shelf yang sesuai dengan standar EPC Kelas-1 Gen-2 [9], yang kemampuan perangkat kerasnya yang terbatas membatasi fungsi yang dapat didukung. Menurut desain kami, sebagian besar kompleksitas protokol ITSP ada di sisi pembaca, tetapi tag juga perlu menyediakan dukungan perangkat keras tertentu. Selain perintah wajib C1G2 (misalnya, Query, Select, dan Read), agar tag dapat mengeksekusi protokol ITSP, kita memerlukan perintah baru yang didefinisikan dalam kumpulan perintah opsional, meminta setiap tag yang terjaga untuk mendengarkan perintah pembaca. menyaring vektor, meng-hash ID-nya ke slot tertentu dari vektor untuk nilai bitnya, tetap diam dan pergi tidur jika nilainya nol, dan merespons dalam slot hash (dengan membuat transmisi untuk membuat saluran sibuk) jika nilainya satu. Perhatikan bahwa tag tidak perlu menyimpan seluruh vektor pemfilteran, tetapi hanya perlu menghitung ke slot yang di-hash, dan mengambil nilai (0/1) yang dibawa dalam slot itu.

Fungsi hash yang efisien perangkat keras dapat ditemukan dalam literatur. Fungsi hash juga dapat diturunkan dari generator nomor pseudo-acak yang diperlukan oleh standar C1G2. Untuk menjaga agar kompleksitas rangkaian tag tetap rendah, kami hanya menggunakan satu fungsi hash yang seragam h(·) dan menggunakannya untuk mensimulasikan beberapa fungsi hash independen: Pada fase pertama dari putaran ke-i, kami menggunakan h(·) dan m_i biji hash unik $\{s_1; s_2; \dots ; s_{mi}\}$ untuk mencapai keluaran hash independen m_i. Dengan demikian, tag id dipetakan ke lokasi bit $(h(id \oplus s_1) \bmod L_{Xi})$, $(h(id \oplus s_2) \bmod L_{Xi})$, $(h(id \oplus s_{mi}) \bmod L_{Xi})$ dalam vektor penyaringan m_i, masing-masing. Setiap benih hash, bersama dengan vektor penyaringan yang sesuai, akan disiarkan ke tag. Pada fase dua putaran ke-i, pembaca menghasilkan benih hash baru s0 dan mengirimkannya ke tag. Setiap tag kandidat di Y_{i+1} memetakan id-nya ke slot indeks $(h(id \oplus s') \bmod L_{Xi})$

2.4 ITSP MELALUI SALURAN BISING

Sejauh ini ITSP mengasumsikan bahwa saluran nirkabel antara pembaca RFID dan tag dapat diandalkan. Perhatikan bahwa protokol CATS juga tidak mempertimbangkan kesalahan saluran. Namun, dalam prakteknya umum bahwa saluran nirkabel jauh dari sempurna karena berbagai alasan, di antaranya kebisingan interferensi dari peralatan terdekat, seperti motor, konveyor, robot, LAN nirkabel, dan telepon nirkabel, adalah salah satu yang penting. Oleh karena itu, bagian ini untuk meningkatkan ITSP dengan membuatnya kuat terhadap gangguan kebisingan.

2.4.1 ITSP dengan Noise on Forward Link

Pembaca mentransmisikan pada tingkat daya yang jauh lebih tinggi dari tag (yang setelah semua backscatter sinyal pembaca dalam kasus tag pasif). Telah ditunjukkan bahwa reader dapat mengirimkan lebih dari satu juta kali lebih tinggi dari *tag backscatte*r. Oleh karena itu, komunikasi tautan maju (*reader* ke *tag*) lebih tahan terhadap gangguan saluran daripada tautan balik (*tag* ke *reader*). Untuk memberikan jaminan tambahan terhadap kebisingan untuk tautan maju, kami dapat menggunakan kode CRC untuk deteksi kesalahan. Standar C1G2 membutuhkan tag untuk mendukung komputasi CRC-16 (16-bit CRC), yang oleh karena itu juga dapat diadopsi oleh tag masa depan yang dimodifikasi untuk ITSP. Setiap vektor pemfilteran yang dibuat oleh pembaca dapat dianggap sebagai kombinasi dari banyak segmen kecil dengan ukuran bit lS yang tetap (misalnya, $l_S = 80$). Untuk setiap segmen, pembaca menghitung CRC 16-bit dan menambahkannya ke akhir segmen itu. Segmen tersebut kemudian digabungkan dan ditransmisikan ke tag. Ketika sebuah tag menerima vektor pemfilteran, pertama-tama tag akan menemukan segmen yang di-*hash* dan menghitung CRC dari segmen tersebut. Jika CRC yang dihitung cocok dengan yang terlampir, ia akan menentukan pencalonannya dengan memeriksa bit di segmen yang dipetakannya. Untuk CRC yang tidak cocok, tag mengetahui bahwa segmen telah rusak, dan akan tetap sebagai tag kandidat terlepas dari nilai bit yang dipetakannya.

Misalkan kita misalkan $l_S = 80$, maka

$$L_{X_i} = \frac{\frac{1}{\ln 2} \times |X_i|}{l_S} \times (l_S + 16) = \frac{1.2|X|}{\ln 2}. \quad (2.31)$$

Kami mengasumsikan probabilitas bahwa noise merusak setiap segmen adalah P_S (P_S diharapkan sangat kecil seperti yang dijelaskan di atas). Segmen yang rusak dapat dianggap terdiri dari semua "1". Oleh karena itu, probabilitas positif palsu untuk vektor penyaringan yang dikirim oleh pembaca, dilambangkan dengan P_{RT}, kira-kira

$$P_{RT} \approx \frac{\frac{L_{X_i}}{96} \times P_S \times l_S + \frac{L_{X_i}}{96} \times (1 - P_S) \times l_S \times P_{FV}}{\frac{L_{X_i}}{96} \times l_S} = \frac{1 + P_S}{2}. \quad (2.32)$$

Kita juga bisa mendapatkan

$$|Y_{i+1}| \approx |V_i| \times (P_{RT})^{m_i} + |W| \quad (2.33)$$

dan sekarang (2.20) dapat ditulis ulang sebagai

$$f(m_i) = \frac{t_s}{\ln 2}\left(1.2m_i|X_i| + \left(\frac{1 + P_{RT}}{2}\right)^{m_i}|V_i| + |W|\right). \quad (2.34)$$

Oleh karena itu, $f(m_i)$ dioptimalkan ketika

$$m_i = \frac{\ln[(\ln 2 - \ln(1 + P_{RT}))|V_i|/1.2|X_i|]}{\ln 2 - \ln(1 + P_{RT})}. \quad (2.35)$$

2.4.2 ITSP dengan Noise pada Tautan Terbalik

Sekarang mari kita pelajari *noise* pada *reverse link* dan pengaruhnya terhadap ITSP. Karena hamburan balik dari tag jauh lebih lemah daripada sinyal yang ditransmisikan oleh pembaca, tautan balik lebih mungkin terpengaruh oleh *noise.*

Pertama, gangguan saluran dapat merusak calon slot kosong menjadi slot sibuk. Slot kosong asli seharusnya diterjemahkan ke dalam bit “0” dalam vektor penyaringan oleh pembaca; jika sebuah kandidat tag dipetakan ke bit itu, itu akan segera dikesampingkan. Namun, jika slot tersebut rusak dan menjadi slot yang sibuk, bit yang sesuai akan berubah menjadi “1”; tag yang dipetakan ke bit itu akan tetap menjadi tag kandidat, sehingga meningkatkan kemungkinan positif palsu dari vektor penyaringan.

Kedua, kebisingan juga dapat terjadi selama slot sibuk. Meskipun derau dan transmisi dari tag dapat membatalkan sebagian satu sama lain dalam slot jika mereka mencapai pembaca dalam fase yang berlawanan, sangat tidak mungkin bahwa mereka akan benar-benar menghilangkan satu sama lain. Selama pembaca masih dapat mendeteksi beberapa energi,

terlepas dari sumbernya (bahkan mungkin berasal dari kebisingan), slot itu akan ditentukan dengan benar sebagai slot sibuk, dan bit yang sesuai dalam vektor penyaringan diatur ke "1" seperti yang seharusnya. Namun, jika kita memperhitungkan rugi-rugi jalur propagasi, termasuk rugi-rugi refleksi, rugi-rugi redaman, dan rugi-rugi penyebaran, masih ada kemungkinan slot yang sibuk mungkin tidak terdeteksi oleh pembaca. Ini mungkin terjadi dalam saluran yang berubah-ubah waktu di mana pembaca mungkin gagal menerima sinyal tag selama slot yang sangat pudar ketika tag mentransmisikan. Kami menekankan bahwa ini bukan masalah yang unik untuk ITSP, tetapi semua protokol yang memerlukan komunikasi dari tag ke pembaca akan mengalami masalah ini jika pembaca tidak dapat mendengar tag. ITSP tidak tahan terhadap jenis kesalahan ini. Tetapi ada cara untuk mengatasi masalah ini misalnya, setiap vektor pemfilteran dari tag ke pembaca ditransmisikan dua kali. Selama slot sibuk di salah satu dari dua transmisi, slot dianggap sibuk. Selanjutnya, kami akan menyelidiki tautan terbalik dengan gangguan kebisingan untuk ITSP di bawah dua model kesalahan.

2.4.2.1 ITSP Berdasarkan Model Kesalahan Acak (ITSP-rem)

Model kesalahan acak dicirikan oleh parameter yang disebut tingkat kesalahan P_{ERR}, yang berarti setiap slot secara independen memiliki kemungkinan P_{ERR} rusak oleh kebisingan. Dipengaruhi oleh kebisingan saluran, pembaca dapat mendeteksi slot yang lebih sibuk karena beberapa slot kosong berubah menjadi slot yang sibuk, yang meningkatkan probabilitas positif palsu dari vektor penyaringan fase-dua. Misalkan ukuran bingkai fase dua dalam putaran tertentu adalah l, jumlah asli slot sibuk adalah sekitar l x $P_{FV} \approx$ l/2. Di sisi pembaca, bagaimanapun, jumlah slot sibuk rata-rata meningkat menjadi l/2 + l/2 x $P_{ERR} = \frac{(1+P_{ERR})x\,1}{2}$. Setelah menyandikan status slot menjadi vektor pemfilteran, probabilitas positif palsu dari vektor pemfilteran itu adalah

$$P'_{FV} \approx \frac{\frac{(1+P_{ERR})\times l}{2}}{l} = \frac{1+P_{ERR}}{2}. \qquad (2.36)$$

Untuk memenuhi persyaratan rasio positif palsu, $(P'_{FV})^K \leq P_{REQ}$ harus berlaku. Oleh karena itu, proses pencarian ITSP-rem setidaknya mengandung putaran.

$$K = \lceil \frac{\ln P_{REQ}}{\ln[(1+P_{ERR})/2]} \rceil \qquad (2.37)$$

Juga, kita dapat memperoleh

$$\begin{aligned} |X_{i+1}| &\approx |U_i| \times P'_{FV} + |W| \\ &\approx |U_i|(1+P_{ERR})/2 + |W|. \end{aligned} \qquad (2.38)$$

Dengan K, $|X_i|$, $|Y_i|$ dan m_i, $1 \leq i \leq K$, waktu pencarian ITSP-rem dapat dihitung menggunakan (2.31) (2.26) (2.27).

2.4.2.2 ITSP Di Bawah Burst Error Model (ITSP-bem)

Dalam telekomunikasi, kesalahan burst didefinisikan sebagai urutan simbol yang diterima secara berurutan, di mana simbol pertama dan terakhir salah, dan tidak ada urutan m yang kontinu (m adalah parameter tertentu yang disebut pita penjaga dari ledakan kesalahan) dengan benar menerima simbol dalam error burst. Model error burst menggambarkan jumlah burst selama interval dan jumlah simbol yang salah di setiap error burst, yang sangat berbeda dari model error acak.

Menurut model error burst yang disajikan pada [6], baik jumlah burst dalam suatu interval maupun jumlah error pada setiap burst memiliki distribusi Poisson. Asumsikan jumlah ledakan yang diharapkan dalam interval l-bit adalah , fungsi distribusi probabilitas untuk jumlah ledakan dapat dinyatakan sebagai

$$h(x) = \sum_{i=0}^{\infty} \frac{\eta^i}{i!} e^{-\eta} \delta_{xi}, \qquad (2.39)$$

di mana δ_{xi} adalah fungsi delta Kronecker. Sedangkan jika nilai rata-rata error akibat burst pada l bit adalah , maka fungsi distribusi probabilitas banyaknya error diberikan oleh

$$g(y) = \sum_{j=0}^{\infty} \frac{\tau^j}{j!} e^{-\tau} \delta_{yj}. \qquad (2.40)$$

Oleh karena itu, probabilitas memiliki w kesalahan dalam interval l bit adalah

$$P_l(w) = e^{-\eta} \frac{\tau^w}{w!} \sum_{i=0}^{\infty} \frac{i^w}{i!} \eta^i e^{-i\tau}. \qquad (2.41)$$

Dengan kata lain, untuk frame dengan slot l, probabilitas bahwa slot w akan rusak oleh noise burst adalah P_l (w).

Sekarang kami mengevaluasi ITSP di bawah model error burst, dilambangkan sebagai ITSP-bem. Diberikan vektor penyaringan dengan ukuran l-bit, ingat dari (2.41) bahwa probabilitas memiliki kesalahan w dalam vektor l-bit ini adalah P_l (w). Dalam hal ini, setiap bit "0" asli memiliki kemungkinan $\frac{w}{l}$ untuk dirusak oleh kesalahan, dan menjadi bit "1". Akibatnya, probabilitas positif palsu dari vektor penyaringan diharapkan menjadi:

$$P'_{FV} \approx \frac{1}{2} + \frac{1}{2} \sum_{w=0}^{l} P_l(w) \times \frac{w}{l}. \qquad (2.42)$$

Setelah mendapatkan nilai P'_{FV}, ITSP-bem dapat menggunakan (2.37), (2.38), untuk menentukan nilai parameter lain yang diperlukan.

2.5 EVALUASI KINERJA

2.5.1 Matrik Kinerja

Kami membandingkan protokol ITSP kami dengan CATS, protokol polling (Bab 2.2.2), DFSA optimal (*dynamic frame slotted* ALOHA), dan protokol identifikasi tag dengan pemulihan tabrakan, dilambangkan sebagai CR, yang mengidentifikasi 4,8 tag per slot rata-rata, sekitar 13 kali kecepatan DFSA optimal. Untuk ITSP dan CATS, filter Bloom mereka (atau vektor penyaringan) merupakan sebagian besar dari keseluruhan transmisi *overhead,* sementara biaya transmisi lainnya, seperti transmisi benih hash, relatif dapat diabaikan. Kedua protokol perlu memperkirakan jumlah tag dalam sistem, |Y|, sebagai langkah pra-protokol. Menurut hasil yang disajikan, waktu untuk memperkirakan |Y| memakan waktu kurang dari 2% dari total waktu eksekusi CATS. Oleh karena itu, kami tidak menghitung waktu estimasi |Y| dalam hasil simulasi karena relatif kecil dan tidak mempengaruhi perbandingan yang adil karena kedua protokol membutuhkannya. Akibatnya, metrik kunci tentang efisiensi waktu adalah ukuran total filter Bloom atau vektor penyaringan, dan kemudian (2.8) dapat digunakan untuk menghitung waktu pencarian yang dibutuhkan oleh CATS, sedangkan (2.27) untuk ITSP.

Setelah proses pencarian selesai, kita akan menghitung rasio positif palsu P_{FP} menggunakan $P_{FP} = \frac{|W* - W|}{|X - W|}$dimana W* adalah himpunan tag dalam hasil pencarian dan W adalah himpunan aktual dari tag yang diinginkan di area cakupan. P_{FP} akan dibandingkan dengan P_{REQ} untuk melihat apakah hasil pencarian memenuhi persyaratan rasio positif palsu.

2.5.2 Perbandingan Kinerja

Kami mengevaluasi kinerja protokol kami dan membandingkannya dengan protokol CATS. Pada set simulasi pertama, kami menetapkan P_{REQ} = 0.001, fix |Y| = 50,000, variasikan |X| dari 5000 hingga 640,000, dan biarkan R_{INTS} = 0,1 , 0,3 , 0,5 , 0,7 , 0,9. Pada set simulasi kedua, kami menetapkan P_{REQ} = 0.001, fix |X| = 10,000, memvariasikan |Y| dari 1250 hingga 40,000 untuk menyelidiki skalabilitas ITSP dengan populasi tag dari rentang yang besar, dan biarkan R_{INTS} = 0,1, 0,3, 0,5, 0,7, 0,9. Untuk kesederhanaan, kami mengasumsikan tid 96ts, dan tl 137ts, di mana perintah QueryAdjust 9-bit atau perintah QueryRep 4-bit, ID 96-bit dan dua nomor acak 16-bit dapat ditransmisikan. Tabel 2.4 dan 2.5 menunjukkan jumlah slot ts yang dibutuhkan oleh protokol di bawah pengaturan parameter yang berbeda. Setiap titik data dalam tabel ini atau gambar/tabel lain di bagian lainnya adalah rata-rata dari 500 simulasi independen yang berjalan dengan kesalahan $\pm$5% atau kurang pada tingkat kepercayaan 95%.

Dari tabel, kami mengamati bahwa ketika R_{INTS} kecil (yang berarti W kecil), kinerja ITSP jauh lebih baik daripada protokol CATS. Misalnya, pada Tabel 2.4, ketika RINTS 0.1, ITSP mengurangi waktu pencarian protokol CATS sebanyak 90,0%. Saat kami meningkatkan RINTS (yang menyiratkan |W| lebih besar), kesenjangan antara kinerja ITSP dan kinerja CATS secara bertahap menyusut.

Tabel 2.4 Perbandingan kinerja protokol pencarian tag. CR berarti protokol identifikasi tag dengan teknik pemulihan tabrakan. |Y| = 50.000, P_{REQ} = 0.001

|X|	***ITSP (R_{INTS})***					***CATS***	***Polling***	***CR***
	0.1	***0.3***	***0.5***	***0.7***	***0.9***			
5,000	61,463	96,989	105,828	108,346	124,553	126,370	485,000	1,427,083
10,000	108,017	145,553	206,709	199,586	231,236	238,313	970,000	1,427,083
20,000	185,204	255,898	335,426	397,462	403,954	447,772	1,940,000	1,427,083
40,000	304,767	467,433	512,156	598,718	678,066	837,837	3,880,000	1,427,083
80,000	414,686	590,150	656,426	721,347	721,347	1,560,259	7,760,000	1,427,083
160,000	472,677	630,669	721,347	721,347	721,347	2,889,689	15,520,000	1,427,083
320,000	529,835	668,794	721,347	721,347	721,347	5,317,715	31,040,000	1,427,083
640,000	573,270	696,015	721,347	721,347	721,347	10,533,732	62,080,000	1,427,083

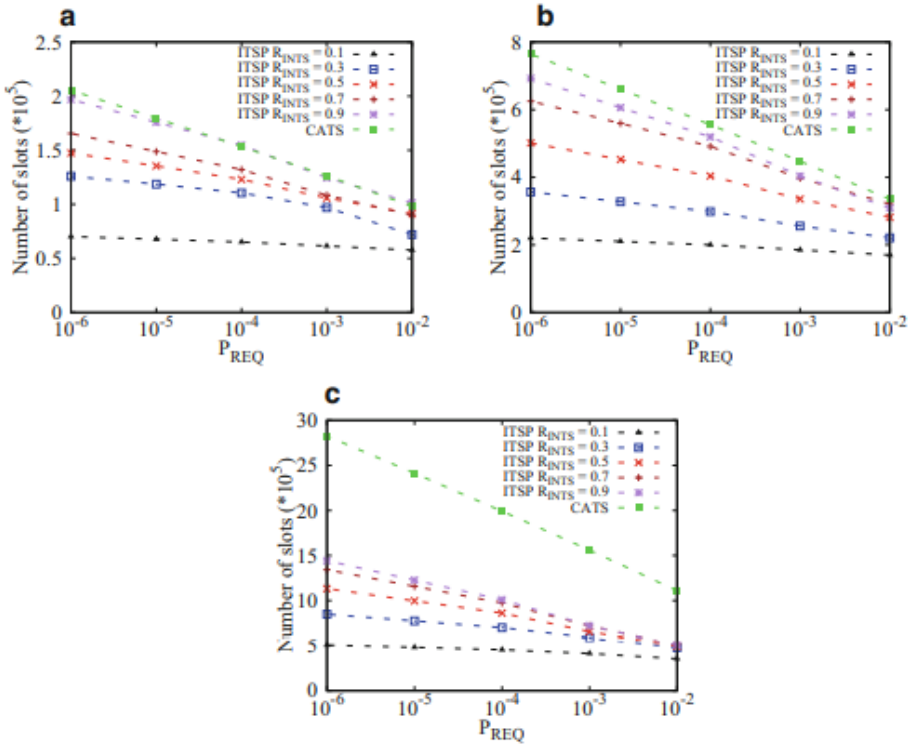
Tabel 2.5 Perbandingan kinerja protokol pencarian tag. CR berarti protokol identifikasi tag dengan teknik pemulihan tabrakan. |X| = 10.000, P_{REQ} = 0.001

|Y|	**ITSP (R_{INTS})**					**CATS**	**Polling**	**CR**
	0.1	**0.3**	**0.5**	**0.7**	**0.9**			
1,250	13,047	17,364	18,033	18,033	18,033	164,589	970,000	35,677
2,500	24,289	33,337	36,067	36,067	36,067	175,960	970,000	71,354
5,000	42,835	62,862	68,528	72,134	72,134	190,387	970,000	142,708
10,000	73,909	109,281	119,022	137,056	144,269	204,814	970,000	285,417
20,000	95,833	132,546	169,065	167,713	192,960	219,241	970,000	570,833
40,000	111,904	152,606	174,926	228,215	232,904	233,668	970,000	1,141,667

Secara khusus, CATS berkinerja buruk ketika $|X| \geq |Y|$. Tetapi ITSP dapat bekerja secara efisien dalam semua kasus. Selain itu, ITSP juga jauh lebih efisien daripada protokol polling, dan protokol identifikasi tag apa pun dengan/tanpa teknik CR. Bahkan dalam kasus terburuk, ITSP hanya membutuhkan sekitar setengah dari waktu eksekusi protokol identifikasi tag dengan teknik CR. (Perhatikan bahwa proses identifikasi sebenarnya membutuhkan lebih banyak waktu karena throughput 4,8 tag per slot mungkin tidak dapat dicapai secara praktis dan durasi setiap slot lebih lama.) Dalam praktiknya, tag yang diinginkan dapat didistribusikan secara spasial di banyak sistem RFID yang berbeda (misalnya, gudang dalam contoh yang kita gunakan dalam pendahuluan), dan dengan demikian RINTS bisa kecil. ITSP adalah protokol yang jauh lebih baik untuk memecahkan masalah pencarian tag dalam skenario praktis ini.

Masalah kinerja lain yang ingin kami selidiki adalah hubungan antara waktu pencarian dan P_{REQ}. Protokol polling, DFSA, dan CR tidak memiliki false positive. Fokus kami akan berada di ITSP dan CATS. Kami menetapkan |X| = 5000, 20.000 atau 80.000, |Y| = 50.000,

memvariasikan R_{INTS} dari 0,1 hingga 0,9, dan memvariasikan P_{REQ} dari 10^{-6} hingga 10^{-2}. Gambar 2.4 membandingkan waktu pencarian yang dibutuhkan oleh CATS dan ITSP di bawah persyaratan rasio positif palsu yang berbeda. Secara umum, gap antara waktu pencarian yang dibutuhkan oleh ITSP dan waktu pencarian oleh CATS semakin besar seiring dengan penurunan P_{REQ}, terutama ketika rintsnya kecil. Sebagai contoh, pada Gambar 2.4c, ketika P_{REQ} 10^{-2} dan R_{INTS} 0.1, waktu pencarian oleh ITSP adalah sekitar sepertiga waktu oleh CATS; ketika kita mengurangi P_{REQ} menjadi 10^{-6}, waktu oleh ITSP menjadi sekitar seperlima dari waktu oleh CATS. Alasannya adalah sebagai berikut: Ketika R_{INTS} kecil, |W| kecil dan sebagian besar tag di |X| dan |Y| bukan kandidat. Setelah beberapa putaran ITSP, karena banyak non-kandidat disaring secara iteratif, ukuran vektor penyaringan berkurang secara eksponensial dan oleh karena itu putaran ITSP berikutnya tidak menyebabkan banyak biaya waktu tambahan. Kelebihan ini membuat ITSP sangat berlaku dalam kasus di mana persyaratan rasio positif palsu sangat ketat, membutuhkan banyak putaran ITSP. Sebaliknya, protokol CATS tidak memiliki kemampuan untuk mengeksploitasi nilai rints yang rendah ini.



Gambar 2.4 Hubungan antara waktu pencarian dan P_{REQ}. Pengaturan parameter: |Y| = 50.000; (a) |X| = 5000, (b) |X| = 20.000, (c) |X| = 80.000

2.5.3 Rasio Positif

Selanjutnya kita kaji apakah hasil pencarian setelah pelaksanaan ITSP memang memenuhi persyaratan P_{REQ}. Dalam simulasi ini, kami menetapkan persyaratan rasio positif palsu berdasarkan rumus berikut:

$$P_{REQ} \leq \frac{|W|}{\lambda\,(|X| - |W|)}, \qquad (2.43)$$

dimana λ adalah konstanta. Kami menggunakan contoh untuk memberikan alasan: Pertimbangkan sistem RFID dengan |X| = 20.000. Jika |W| = 10.000, P_{REQ} = 0.01 mungkin cukup baik karena jumlah positif palsu sekitar (|X| - |W|) x P_{REQ} = 100, yang jauh lebih sedikit daripada |W|. Namun, jika |W| = 10, P_{REQ} = 0.01 dapat menjadi tidak dapat diterima karena (|X| - |W|) x $P_{REQ} \approx 200 \gg |W|$. Oleh karena itu, diinginkan untuk mengatur nilai P_{REQ} sedemikian rupa sehingga jumlah positif palsu dalam hasil pencarian jauh lebih kecil dari |W|, yaitu (|X| - |W|) x $P_{REQ} \leq \frac{1}{\lambda}$ |W|. Misalkan λ = 10 dan kita uji ITSP di bawah tiga pengaturan parameter yang berbeda:

1. |X| = 5000, |Y| = 50.000 dam R_{INTS} bervariasi dari 0.1 hingga 0.9 yaitu |W| bervariasi dari 500 hingga 4500. $P_{REQ} \leq \frac{500}{10\,x\,(5000-500)} \approx 0.01111$. Kita menetapkan $P_{REQ} = 10^{-2}$.
2. |X| = 20.000, |Y| = 50.000 dam R_{INTS} bervariasi dari 00.1 hingga 0.9 yaitu |W| bervariasi dari 200 hingga 18.000. $P_{REQ} \leq \frac{200}{10\,x\,(20.000-200)} \approx 0.00101$. Kita menetapkan $P_{REQ} = 10^{-3}$.
3. |X| = 80.000, |Y| = 50.000 dam R_{INTS} bervariasi dari 0.01 hingga 0.9 yaitu |W| bervariasi dari 500 hingga 45.000. $P_{REQ} \leq \frac{500}{10\,x\,(80.000-500)} \approx 0.00063$. Kita menetapkan $P_{REQ} = 10^{-4}$.

Untuk setiap pengaturan parameter, kami mengulangi simulasi 500 kali untuk mendapatkan rasio positif palsu rata-rata. Gambar 2.5 menunjukkan hasil simulasi. Dalam (a), (b), dan (c), kita dapat melihat bahwa P_{FP} rata-rata selalu lebih kecil dari P_{REQ} yang sesuai. Oleh karena itu, hasil pencarian menggunakan ITSP memenuhi persyaratan rasio positif palsu yang ditentukan dalam arti rata-rata.

Jika kita melihat ke dalam rincian simulasi individu, kita menemukan bahwa sebagian kecil dari simulasi berjalan memiliki P_{FP} di luar P_{REQ}. Misalnya, Gambar 2.6 menggambarkan hasil 500 run dengan |X| = 5000, |Y| = 50.000, |W| = 500, dan P_{REQ} 10^{-2}. Ada sekitar 5% run yang memiliki $P_{FP} > P_{REQ}$, tetapi itu tidak mengejutkan karena rasio positif palsu dalam konteks vektor penyaringan (ITSP) atau filter Bloom (CATS) didefinisikan dengan cara probabilitas: Probabilitas untuk setiap tag di X - W yang salah diklasifikasikan sebagai satu di W tidak lebih besar dari P_{REQ}. Definisi probabilistik ini memberlakukan persyaratan P_{REQ} dalam arti rata-rata, tetapi tidak mutlak untuk setiap proses individu.

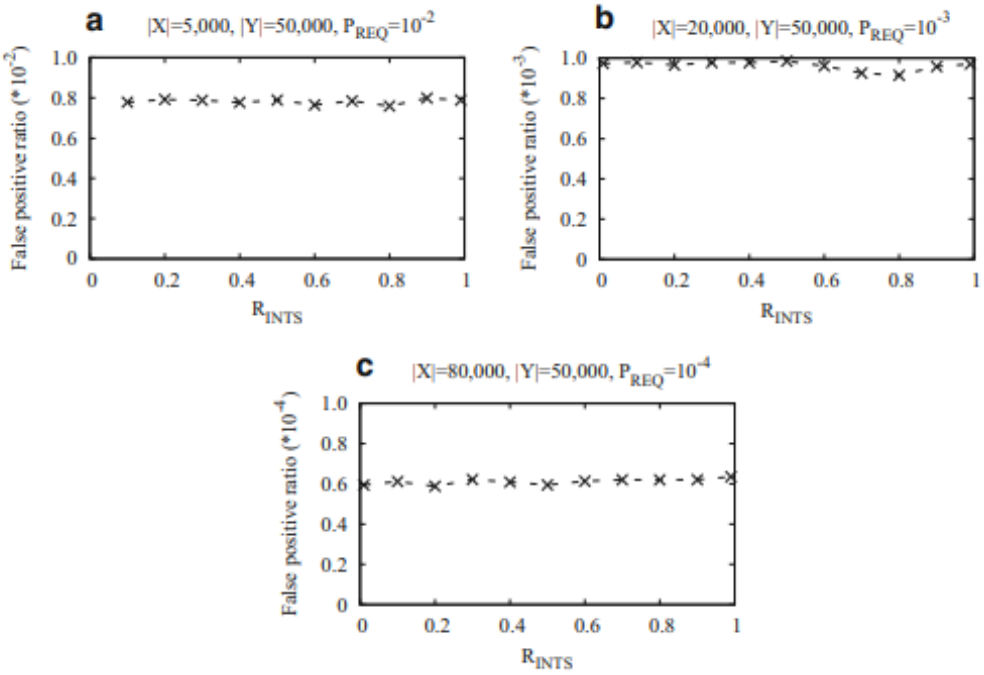
2.5.4 Evaluasi Kinerja Di Bawah Kesalahan Saluran

2.5.4.1 Kinerja ITSP-rem dan ITSP-bem

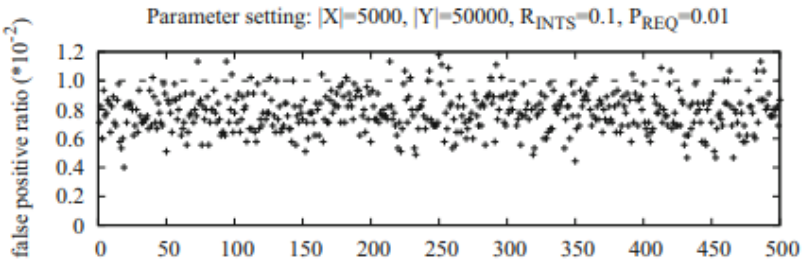
Kami mengevaluasi kinerja ITSP-rem dan ITSP-bem. Untuk mensimulasikan tingkat kesalahan P_{ERR} di ITSP-rem, kami menggunakan generator nomor pseudo-acak, yang menghasilkan bilangan real acak yang seragam dalam kisaran [0, 1]. Jika bit dalam vektor pemfilteran adalah “0” dan bilangan acak yang dihasilkan adalah [0, P_{ERR}], bit itu

dibalik ke "1". P_S dapat disimulasikan dengan cara yang serupa. Sedangkan untuk burst error pada ITSP-bem, terlebih dahulu kita hitung nilai P_l (w) dengan w yang berbeda untuk l yang diberikan. Kemudian setiap w ditetapkan dengan rentang yang tidak tumpang tindih di [0, 1], yang panjangnya sama dengan nilai P_l (w). Untuk setiap interval, kami menghasilkan nomor acak dan memeriksa rentang mana yang ditemukan oleh nomor tersebut, sehingga menentukan jumlah kesalahan dalam interval tersebut.

Kami menetapkan P_{REQ} = 0.001, P_S = 0,01, dan R_{INTS} 0.1, 0.5, 0.9, masing-masing. Nilai X dan Y sama dengan yang ada pada Tabel 2.4 dan 2.5. ls diatur ke 80 bit dan CRC 16-bit ditambahkan ke setiap segmen pada tautan maju untuk pemeriksaan integritas. Untuk ITSP-rem, kami mempertimbangkan dua kasus dengan PERR = 5% dan 10%, masing-masing. Untuk ITSP-bem, parameter yang ditentukan diatur menjadi: 0.135, 7.10 dengan setiap interval menjadi 96 bit.



Gambar 2.5 Rasio positif palsu setelah menjalankan ITSP



Gambar 2.6 Rasio positif palsu oleh ITSP dari 500 run

Tabel 2.6 Perbandingan kinerja. |Y| = 50.000, R_{INTS} = 0.1, P_{REQ} = 0.001

|X|	**ITSP**	**ITSP-rem**		**ITSP-bem**
		P_{ERR} **D 5 %**	P_{ERR} **D 10 %**	
5,000	61,463	74,288	75,812	72,144
10,000	108,017	129,995	133,022	125,779
20,000	185,204	241,026	247,824	238,962
40,000	304,767	361,242	398,198	358,361
80,000	414,686	441,365	458,433	437,256
160,000	472,677	504,565	545,338	499,058
320,000	529,835	567,403	630,174	560,456
640,000	573,270	626,379	690,400	618,913

Tabel 2.7 Perbandingan kinerja. . |Y| = 50.000, R_{INTS} = 0.5, P_{REQ} = 0.001

|X|	**ITSP**	**ITSP-rem**		**ITSP-bem**
		P_{ERR} **D 5 %**	P_{ERR} **D 10 %**	
5,000	105,828	160,481	166,469	153,838
10,000	206,709	211,513	221,771	210,805
20,000	335,426	371,974	391,983	370,557
40,000	512,156	577,305	617,196	577,305
80,000	656,426	735,592	789,874	735,592
160,000	721,347	793,482	865,617	793,482
320,000	721,347	793,482	865,617	793,482
640,000	721,347	793,482	865,617	793,482

Tabel 2.6, 2.7, 2.8, 2.9, 2.10, dan 2.11 menunjukkan jumlah slot ts yang dibutuhkan pada setiap pengaturan parameter. Kolom kedua menyajikan hasil ITSP ketika saluran benar-benar andal. Kolom ketiga dan keempat menyajikan hasil ITSP-rem dengan tingkat kesalahan 5% atau 10%. Kolom kelima menyajikan hasil ITSP-bem. Tidak mengherankan bahwa proses pencarian di bawah saluran yang bising umumnya membutuhkan lebih banyak waktu karena penggunaan CRC dan probabilitas positif-palsu yang lebih tinggi dari vektor penyaringan, dan waktu pelaksanaan ITSP-rem biasanya lebih lama di saluran dengan frekuensi yang lebih tinggi. Tingkat kesalahan. Pengamatan positif yang penting adalah bahwa kinerja ITSP menurun dengan baik di semua simulasi. Peningkatan waktu eksekusi baik untuk ITSP-rem maupun ITSP-bem tergolong sederhana, dibandingkan dengan ITSP dengan saluran yang sempurna. Misalnya, bahkan ketika tingkat kesalahan 10%, waktu eksekusi ITSP-rem sekitar 10–

30% lebih tinggi daripada ITSP. Peningkatan sederhana ini menunjukkan kepraktisan protokol kami di bawah saluran yang bising.

2.5.4.2 Rasio Positif Palsu ITSP-rem dan ITSP-bem

Kami menggunakan pengaturan parameter yang sama di Sect. 2.5.3 untuk menguji keakuratan hasil pencarian oleh ITSP-rem dan ITSP-bem. Sedangkan untuk ITSP-rem kita tetapkan P_{ERR} = 5% atau 10%. Untuk ITSP-bem, pengaturan parameter input yang diperlukan adalah η = 0.135 dan τ = 7.10, dengan setiap interval 96-bit. Hasil simulasi digambarkan pada Gambar 2.7,

Tabel 2.8 Perbandingan kinerja |Y| = 50.000, R_{INTS} = 0.9, P_{REQ} = 0.001

|X|	*ITSP*	*ITSP-rem*		*ITSP-bem*
		***P*ERR D 5 %**	***P*ERR D 10 %**	
5,000	124,553	156,041	163,718	155,972
10,000	231,236	275,394	290,493	275,256
20,000	403,954	454,929	486,150	454,929
40,000	678,066	752,753	814,890	752,753
80,000	721,347	793,482	865,617	793,482
160,000	721,347	793,482	865,617	793,482
320,000	721,347	793,482	865,617	793,482
640,000	721,347	793,482	865,617	793,482

Tabel 2.9 Perbandingan kinerja. |X| = 10.000, R_{INTS} = 0.1, P_{REQ} = 0.001

|Y|	**ITSP**	**ITSP-rem**		**ITSP-bem**
		***P*ERR D 5 %**	***P*ERR D 10 %**	
1,250	13,047	14,868	15,898	14,174
2,500	24,289	26,626	28,617	25,283
5,000	42,835	46,994	50,863	44,393
10,000	73,909	76,807	84,135	75,983
20,000	95,833	103,255	106,693	102,121
40,000	111,904	133,043	137,348	130,382

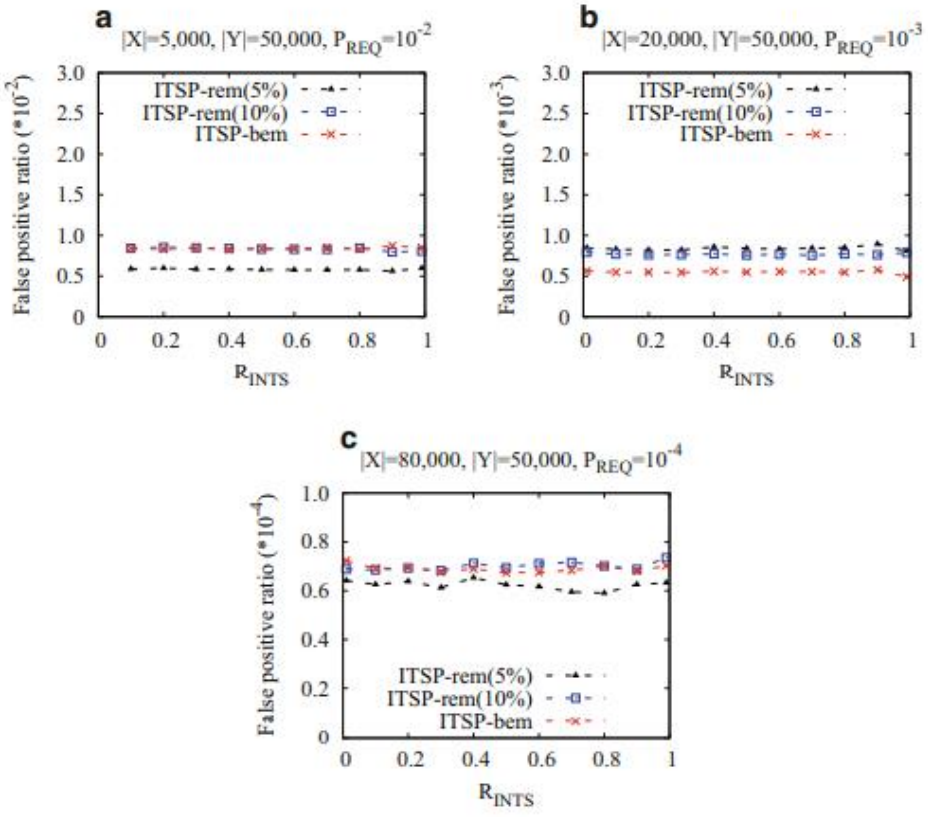
Tabel 2.10 Perbandingan kinerja. |X| = 10.000, R_{INTS} = 0.5, P_{REQ} = 0.001

|Y|	**ITSP**	**ITSP-rem**		**ITSP-bem**
		***P*ERR D 5 %**	***P*ERR D 10 %**	
1,250	18,033	19,837	21,640	19,837

2,500	36,067	39,674	43,280	39,674
5,000	68,528	77,021	82,448	77,021
10,000	119,022	134,208	143,261	134,208
20,000	169,065	202,891	212,105	202,467
40,000	174,926	214,563	224,227	213,970

Tabel 2.11 Perbandingan kinerja. |X| = 10.000, R_{INTS} = 0.9, P_{REQ} = 0.001

|Y|	ITSP	ITSP-rem		ITSP-bem
		P_{ERR} D 5 %	P_{ERR} D 10 %	
1,250	18,033	19,837	21,640	19,837
2,500	36,067	39,674	43,280	39,674
5,000	72,134	79,348	86,561	79,348
10,000	144,269	158,696	173,123	158,696
20,000	192,960	217,245	232,272	217,245
40,000	232,904	261,277	277,300	261,173

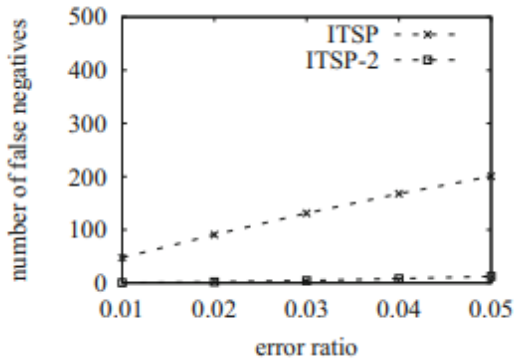


Gambar 2.7 Rasio positif palsu setelah menjalankan ITSP-rem, ITSP-bem, dan CATS. (a) |X| = 5000, |Y| = 50.000, $P_{REQ} = 10^{-2}$, (b) |X| =20.000, |Y| =50.000, P_{REQ} D 10^{-3}, (c) |X| =80.000, |Y| = 50.000, $P_{REQ} = 10^{-4}$

di mana tingkat kesalahan diberikan di antara tanda kurung setelah ITSP-bem. Jelas, rasio positif palsu dalam hasil pencarian setelah mengeksekusi ITSP-rem atau ITSP-bem selalu dalam batas PREQ. Hasil ini mengkonfirmasi bahwa persyaratan rasio positif palsu terpenuhi di bawah saluran yang bising.

2.5.4.3 Sinyal Hilang Karena Saluran Memudar

Kami mempertimbangkan skenario saluran yang bervariasi waktu di mana mungkin terjadi bahwa sinyal dari tag tidak diterima oleh pembaca di slot fading yang dalam. Meskipun kami menganggap kondisi ini relatif jarang terjadi pada sistem RFID yang dikonfigurasi untuk bekerja secara stabil, kami mengakui di Sect. 2.4.2 bahwa ITSP (atau CATS) tidak tahan terhadap jenis kesalahan ini. Namun, masalahnya dapat diatasi dengan tag yang mentransmisikan setiap vektor penyaringan dua kali. Gambar 2.8 menunjukkan hasil simulasi di bawah parameter |X| = 10000, |Y| = 5000, |W| = 500, dan P_{REQ} = 0.01.



Gambar 2.8 Negatif palsu karena hilangnya sinyal pada saluran yang berubah-ubah waktu

Sumbu horizontal menunjukkan tingkat kesalahan, yang didefinisikan sebagai fraksi slot dalam fading dalam, yang menyebabkan hilangnya sinyal sepenuhnya. ITSP-2 menunjukkan pendekatan transmisi setiap vektor penyaringan dari tag ke pembaca dua kali. Ketika tag yang diinginkan di W tidak diidentifikasi, kami menyebutnya negatif palsu. Hasil simulasi menunjukkan bahwa ITSP menimbulkan negatif palsu yang signifikan ketika tingkat kesalahan menjadi besar. Misalnya, ketika tingkat kesalahan adalah 2%, jumlah rata-rata negatif palsu adalah 90,7. ITSP-2 bekerja sangat baik dalam mengurangi angka ini. Ketika tingkat kesalahan adalah 2%, jumlah negatif palsunya hanya 1,95.

2.6 RINGKASAN

Bab ini membahas masalah pencarian tag dalam sistem RFID skala besar. Kami menghadirkan protokol pencarian tag berulang (ITSP) yang meningkatkan efisiensi waktu dan menghilangkan batasan solusi sebelumnya. Selain itu, kami memperluas ITSP untuk bekerja di bawah saluran yang bising. Kontribusi utama dari pekerjaan kami dirangkum sebagai berikut: (1) Metode iteratif ITSP berdasarkan vektor penyaringan sangat efektif dalam mengurangi jumlah informasi yang akan dipertukarkan antara tag dan pembaca, dan akibatnya menghemat waktu dalam proses pencarian; (2) ITSP berkinerja jauh lebih baik daripada solusi yang ada; (3) ITSP bekerja dengan baik di bawah semua kondisi sistem, terutama dalam situasi $|X| \gg |Y|$ ketika CATS bekerja dengan buruk; (4) ITSP ditingkatkan untuk bekerja secara efektif di bawah saluran yang bising.

BAB 3
OTENTIKASI RFID

Bab ini menjelaskan tentang otentikasi anonim RFID yang ringan. Meluasnya penggunaan tag RFID menimbulkan masalah privasi: Mereka membuat operator mereka dapat dilacak. Untuk melindungi privasi pembawa tag, kita perlu menemukan mekanisme baru yang menjaga kegunaan tag saat melakukannya secara anonim. Banyak aplikasi tag seperti pembayaran tol memerlukan otentikasi. Karena tag berbiaya rendah memiliki sumber daya perangkat keras yang sangat terbatas, prinsip desain asimetris diadopsi untuk mendorong sebagian besar kompleksitas ke pembaca RFID yang lebih kuat. Alih-alih menerapkan fungsi hash kriptografik yang rumit dan intensif perangkat keras, protokol autentikasi kami hanya memerlukan tag untuk melakukan beberapa operasi sederhana dan efisien perangkat keras guna menghasilkan token dinamis untuk autentikasi anonim. Analisis teoretis dan uji keacakan menunjukkan bahwa protokol kami dapat memastikan privasi tag. Selain itu, protokol kami mengurangi overhead komunikasi dan overhead komputasi online ke O.1/ per otentikasi untuk tag dan pembaca, yang lebih baik dibandingkan dengan teknologi sebelumnya.

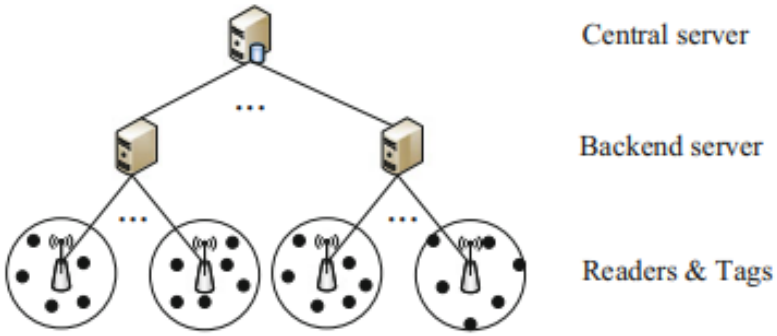
Sisa bab ini disusun sebagai berikut. Bagian 3.1 menjelaskan model sistem dan model keamanan. Bagian 3.2 memberikan pekerjaan terkait. Bagian 3.3–3.5 menyajikan tiga protokol otentikasi RFID anonim. Analisis keamanan dilakukan di Sect. 3.6. Bagian 3.7 memberikan evaluasi numerik. Bagian 3.8 memberikan ringkasan.

3.1 Model Sistem dan Model Keamanan

3.1.1 Model Sistem

Pertimbangkan sistem RFID terdistribusi hirarkis seperti yang ditunjukkan pada Gambar. 3.1. Setiap tag sudah diinstal sebelumnya dengan beberapa kunci untuk otentikasi. Pembaca ditempatkan di lokasi yang dipilih, bertanggung jawab untuk mengautentikasi tag yang memasuki area jangkauan mereka. Selain itu, pembaca di setiap lokasi terhubung ke server backend, berfungsi sebagai suplemen untuk menyediakan lebih banyak sumber daya penyimpanan dan komputasi. Semua server backend selanjutnya terhubung ke server pusat, di mana setiap kunci tag disimpan. Server backend resmi mana pun dapat mengambil kunci tag dari server pusat. Karena kunci dari setiap tag hanya disimpan di server pusat, kunci tersebut disinkronkan dari tampilan server backend yang berbeda. Selain itu, tautan berkecepatan tinggi yang menghubungkan server pusat, server backend, dan pembaca membuat latensi transmisi data otentikasi kecil dapat diabaikan. Oleh karena itu, pembaca, server backend yang terhubung, dan server pusat dapat dianggap sebagai satu kesatuan, dan akan digunakan secara bergantian.

Dalam bab ini, kami fokus pada tag RFID berbiaya rendah, khususnya tag hamburan balik pasif yang banyak digunakan saat ini. Kesederhanaan tag ini berkontribusi pada harga rendahnya, yang pada gilirannya membatasi kemampuan komputasi, komunikasi, dan penyimpanannya. Sebaliknya, pembaca, yang tidak dibutuhkan dalam jumlah besar seperti tag, dapat memiliki sumber daya yang jauh lebih kaya. Selain itu, server backend dapat memberi pembaca sumber daya tambahan bila diperlukan. Komunikasi antara pembaca dan tag bekerja dalam mode permintaan-dan-tanggapan. Pembaca memulai komunikasi dengan mengirimkan permintaan. Setelah menerima permintaan, tag membuat transmisi yang sesuai sebagai tanggapan. Kami membagi transmisi antara pembaca dan tag menjadi dua jenis: (1) Transmisi invarian berisi konten yang tidak berubah antara tag apa pun dan pembaca mana pun, seperti transmisi suar dari pembaca yang menginformasikan tag masuk tentang apa yang harus dilakukan selanjutnya. (2) Transmisi varian berisi konten yang dapat bervariasi untuk tag yang berbeda atau tag yang sama pada waktu yang berbeda, seperti data yang dipertukarkan untuk otentikasi anonim.



Gambar 3.1 Sistem RFID terdistribusi secara hierarkis

3.1.2 Model Keamanan

Model Ancaman. Musuh dapat menguping transmisi nirkabel apa pun yang dibuat antara tag dan pembaca. Selain itu, musuh dapat memasang pembaca yang tidak sah di lokasi yang dipilih, yang berkomunikasi dengan tag yang lewat dan mencoba mengidentifikasi pembawa tag. Namun, pembaca yang tidak sah tersebut tidak memiliki akses ke server backend atau server pusat karena server akan mengautentikasi pembaca sebelum memberikan izin akses. Dalam sekuelnya, pembaca tanpa notasi lebih lanjut berarti yang diotorisasi secara default. Selain itu, kami berasumsi bahwa musuh dapat mengkompromikan beberapa tag dan mendapatkan kunci mereka, tetapi itu tidak dapat membahayakan pembaca resmi mana pun.

Model Anonim. Model anonim mensyaratkan bahwa semua transmisi varian harus tidak dapat dibedakan oleh musuh, yang berarti bahwa (1) transmisi varian apa pun dalam protokol tidak boleh membawa nilai tetap yang tidak berubah di beberapa otentikasi, dan (2) konten transmisi akan tampak benar-benar acak dan tidak terkait di seluruh autentikasi yang berbeda dengan penyadap yang menangkap transmisi. Oleh karena itu, tidak ada musuh yang memiliki

keunggulan yang tidak dapat diabaikan dalam berhasil menebak transmisi varian berikutnya dari tag berdasarkan transmisi sebelumnya. Notasi yang digunakan dalam bab ini diberikan pada Tabel 3.1 untuk referensi cepat.

Tabel 3.1 Notasi

Simbol	Deskripsi
n	Jumlah tag dalam sistem RFID terdistribusi
n_t	Jumlah total token yang digunakan oleh tag
R	Pembaca RFID
t	Sebuah tag RFID
idx	Indeks tag ($1 \leq i \leq n$)
tk	Token untuk otentikasi
ic	Sebuah indikator
$tk[i]$	Bit ke-i dalam token tk
$ic[i]$	Bit ke-i di ic indikator
t_i	Tag ke-i ($1 \leq i \leq n$) dalam sistem
KT	Tabel kunci disimpan dalam database
$KT[i]$	Kunci dari tag t_i disimpan di tabel kunci
tk^j_i	Token ke-j disimpan oleh tag t_i
pt_i	Indeks token disimpan oleh tag t_i
u	Jumlah token dasar yang disimpan oleh setiap tag
v	Jumlah indikator dasar yang disimpan oleh setiap tag
Bt^j_i	Token dasar ke-j ($1 \leq j \leq u$) dari tag t_i
Bi^j_i	Indikator dasar ke-j ($1 \leq j \leq v$) dari tag t_i
ic_i	Indikator tag saat ini
a	Panjang setiap token/token dasar
b	Panjang setiap indikator/indikator dasar
ρ	Rasio pemanfaatan memori dari tabel hash

3.2 PEKERJAAN TERKAIT

Pekerjaan sebelumnya pada otentikasi anonim secara umum dapat diklasifikasikan ke dalam dua kategori: berbasis non-pohon dan berbasis pohon.

3.2.1 Protokol Berbasis Non-pohon

Kunci hash memanfaatkan nilai hash acak untuk otentikasi anonim. Setelah menerima permintaan otentikasi dari pembaca, sebuah tag mengirim kembali (r; id $\oplus f_k(r)$), di mana r adalah bilangan acak, id adalah ID tag, k adalah rahasia yang dibagikan sebelumnya antara tag dan pembaca, dan $\{f_n\}_n \in \mathbb{N}$ adalah ansambel fungsi bilangan acak semu. Pembaca secara mendalam mencari database untuk tag yang ID dan kuncinya dapat menghasilkan kecocokan

dengan data yang diterima. Protokol hash-lock memiliki masalah efisiensi yang serius sehingga pembaca perlu melakukan perhitungan hash O.n/ on line per otentikasi, di mana n adalah jumlah tag dalam sistem. Beberapa varian skema hash-lock mencoba meningkatkan efisiensi pencarian, tetapi mereka memiliki masalah. Protokol OKS menggunakan rantai hash untuk otentikasi anonim. Protokol OSK/AO memanfaatkan tradeoff waktu-memori untuk mengurangi pencarian 2 2 kompleksitas ke O($n^{\frac{2}{3}}$) (masih terlalu besar) dengan biaya O($n^{\frac{2}{3}}$) unit memori. Namun, baik OKS dan OSK/AO tidak dapat menjamin anonimitas di bawah serangan denial-of-service (DoS). Protokol YA-TRAP memanfaatkan stempel waktu yang meningkat secara monoton untuk mencapai otentikasi anonim. YA-TRAP juga rentan terhadap serangan DoS, dan sebuah tag hanya dapat diautentikasi sekali dalam setiap unit waktu. Serangan DoS di OSK/AO dan YA-TRAP pada dasarnya adalah serangan desinkronisasi, yang mengelabui tag agar memperbarui kuncinya secara tidak perlu dan membuatnya gagal untuk diautentikasi oleh pembaca resmi nanti.

Protokol TERAKHIR dirancang berdasarkan model privasi yang lemah. Pengidentifikasi kunci digunakan untuk memudahkan pembaca mengidentifikasi tag dengan cepat. Setelah setiap otentikasi, pembaca mengunggah pengidentifikasi baru; pasangan kunci ke tag. LAST hanya memerlukan pembaca dan tag untuk menghitung O/1 hash per otentikasi, tetapi overhead pembaca untuk mencari pengidentifikasi kunci yang diberikan tidak dipertimbangkan. Selain itu, karena pengidentifikasi kunci hanya diperbarui setelah otentikasi berhasil, tag terus mengirimkan pengidentifikasi kunci yang sama antara dua otentikasi sukses berturut-turut. Oleh karena itu, LAST bukanlah anonim dalam arti yang sebenarnya. Selain itu, proses mengunggah pengenal baru; pasangan kunci ke tag setelah setiap otentikasi menimbulkan overhead komunikasi tambahan.

3.2.2 Protokol Berbasis Pohon

Protokol berbasis pohon mengatur kunci bersama dalam pohon seimbang untuk mengurangi kerumitan dalam mengidentifikasi tag ke O(log n). Namun, protokol berbasis pohon umumnya mengharuskan setiap tag untuk menyimpan kunci O(log n), yaitu O(1) untuk protokol berbasis non-pohon.

Dalam protokol Dimitriou, node non-daun dari pohon menyimpan kunci tambahan yang dapat digunakan untuk menyimpulkan jalur menuju node daun yang menyimpan kunci otentikasi. Untuk setiap autentikasi, overhead komputasi untuk pembaca dan tag adalah O(log n), dan tag perlu mentransmisikan nilai hash O(log n). Protokol ini rentan terhadap serangan kompromi karena tag yang berbeda dapat berbagi kunci tambahan.

Protokol ECNP memanfaatkan teknik pengkodean kriptografi untuk mengompresi data otentikasi yang dikirimkan oleh tag. ECNP dapat mengurangi overhead komputasi pembaca dan overhead transmisi tag dengan berlipat ganda dibandingkan dengan protokol Dimitriou, tetapi mereka tetap O.log n/ karena penggunaan struktur pohon. Selain itu, ECNP tidak tahan

terhadap serangan kompromi karena anak-anak dari satu simpul di pohon berbagi kunci grup yang sama.

Protokol ACTION dirancang agar tahan terhadap serangan yang membahayakan. Ini mengadopsi arsitektur pohon jarang untuk membuat kunci dari setiap tag independen satu sama lain. Dalam ACTION, setiap tag secara acak ditetapkan dengan kunci jalur, yang selanjutnya disegmentasi ke dalam indeks tautan untuk memandu pembaca berjalan menuruni pohon menuju simpul daun yang membawa kunci rahasia k dari tag. Untuk setiap otentikasi, sebuah tag perlu menghitung dan mengirimkan hash O(log n) dan pembaca perlu melakukan O(log n) hash untuk menemukan kunci bersama. Masalah utama ACTION adalah ukuran indeks tautan terlalu kecil setelah segmentasi (misalnya, 4 bit), membuatnya mudah ditebak.

3.3 SOLUSI STRAWMAN

Kami pertama-tama memperkenalkan solusi strawman untuk otentikasi anonim ringan menggunakan token yang telah disimpan sebelumnya.

3.3.1 Motivasi

Sebagian besar wajan sebelumnya, jika tidak semuanya, menggunakan fungsi hash kriptografik untuk mengacak data otentikasi dengan tujuan menjaga anonimitas. Menerapkan fungsi hash kriptografi khas seperti MD4, MD5, dan SHA-1 membutuhkan setidaknya gerbang logika 7K. Namun, tag pasif yang banyak digunakan hanya memiliki gerbang logika 7K–15K, di mana 2K–5K dicadangkan untuk tujuan keamanan. Kendala perangkat keras memerlukan paradigma desain baru untuk protokol otentikasi anonim ringan yang lebih mendukung untuk tag berbiaya rendah. Keberhasilan komersial tag RFID terletak pada kesederhanaannya. Meskipun tidak ada spesifikasi tentang seberapa sederhana tag ini, aman untuk mengatakan bahwa kami akan selalu memilih solusi yang mencapai tujuan yang sebanding dengan kebutuhan perangkat keras yang lebih sedikit. Di sisi lain, perbedaan yang signifikan antara pembaca dan tag menunjukkan prinsip desain asimetri yang akan kami ikuti: mendorong sebagian besar kompleksitas ke pembaca sambil membiarkan tag sesederhana mungkin.

3.3.2 Solusi Strawman

Pertimbangkan sistem RFID dengan n tag t_1, t_2, … , t_n, masing-masing pra-instal dengan array m token acak unik, [tk_i^1, tk_i^2, … , tk_i^m] ($1 \leq i \leq n$). Tag ti juga memiliki indeks token pti (diinisialisasi ke 1) yang menunjuk ke token pertama yang tidak digunakan. Token dan indeks token dari setiap tag juga disimpan dalam database server pusat, seperti yang diilustrasikan pada Tabel 3.2.

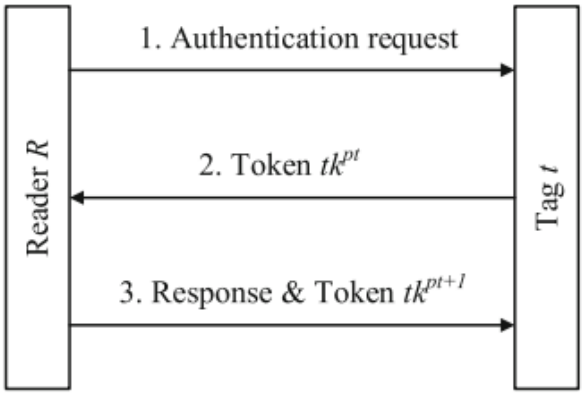
Dalam sekuelnya, kami mempertimbangkan proses otentikasi antara pembaca arbitrer R dan tag arbitrer t yang memiliki array token [tk^1, tk^2, … , tk^m] dan indeks token pt. Untuk mengotentikasi t, R mengirimkan permintaan ke t. Setelah menerima permintaan, t mengirimkan token tk^{pt} pertama yang tidak digunakan ke R. Setelah itu, t meningkatkan pt sebesar 1 untuk menjamin bahwa token yang sama tidak akan digunakan dua kali. Jika tidak, t

akan selalu mengirim token yang sama ketika pembaca yang tidak sah meminta token, yang merusak anonimitas. Setelah menerima token, R harus mencari token di database karena tidak tahu identitas tag. Mulai dari i = 1, R memeriksa apakah $tk^{pt}_i = tk^{pt}$ satu per satu. Jika ada $i \in [1,n]$ sehingga $tk^{pt}_i = tk^{pt}$, t berhasil diautentikasi; jika tidak, t gagal otentikasi. Dalam kasus sebelumnya, R mengirimkan kembali token tkptiC1 ke t untuk mengotentikasi dirinya sendiri, dan menetapkan $pt_i = pt_i + 2$. Tag t membandingkan token yang diterima dengan tk^{pt} untuk mengotentikasi R, dan meningkatkan pt sebanyak 1 lagi. Gambar 3.2 menunjukkan tiga langkah otentikasi bersama.

Dalam pendekatan ini, biaya komputasi online tag rendah—hanya satu perbandingan (memerlukan perangkat keras yang jauh lebih sedikit daripada penerapan hash kriptografis) per autentikasi. Kompleksitas komputasi online pembaca adalah O(n) karena paling banyak n perbandingan (meskipun satu perbandingan jauh lebih murah daripada menghitung satu nilai hash) diperlukan untuk mencari token yang diterima. Overhead komunikasi untuk pembaca dan tag adalah O(1).

Tabel 3.2 Tabel kunci untuk desain awal

Tag	Token array	Token index
t_1	$[tk_1^1, tk_1^2, \dots, tk_1^m]$	pt_1
t_2	$[tk_2^1, tk_2^2, \dots, tk_2^m]$	pt_2
⋮	⋮	⋮
t_n	$[tk_n^1, tk_n^2, \dots, tk_n^m]$	pt_n



Gambar 3.2 Tiga langkah otentikasi timbal balik berbasis token

Untuk menghindari kebocoran identitas tag, token yang digunakan untuk otentikasi harus terlihat acak. Selain itu, setiap token hanya dapat digunakan satu kali. Oleh karena itu, tag harus diisi ulang dengan token baru setelah $\frac{m}{2}$ otentikasi timbal balik, misalnya membeli token baru dari cabang resmi. Oleh karena itu, tag harus menyimpan token sebanyak mungkin

untuk mengurangi ketidaknyamanan yang disebabkan oleh pengisian token. Namun, tag berbiaya rendah hanya memiliki memori kecil. Misalnya, tag UHF pasif umumnya memiliki memori pengguna 512-bit untuk menyimpan data khusus pengguna (token dalam kasus kami). Beberapa tag kelas atas dengan memori besar sangat mahal untuk diterapkan dalam jumlah besar. Sebagai contoh, x Sky-ID tag dengan memori pengguna 8 KB masing-masing berharga Rp 375.000 . Kami akan memperkenalkan masalah keamanan desain ini di bagian selanjutnya.

3.4 PROTOKOL OTENTIKASI BERBASIS TOKEN DINAMIS

Di bagian ini, kami menjelaskan Protokol *Otentikasi Berbasis Token* (TAP) dinamis pertama kami. TAP dapat menghasilkan token untuk otentikasi anonim sesuai permintaan, dan oleh karena itu tidak memerlukan tag untuk menginstal banyak token terlebih dahulu. Namun, kami akan segera menunjukkan bahwa TAP masih memiliki beberapa masalah, yang akan diselesaikan dengan desain akhir kami di bagian selanjutnya.

3.4.1 Motivasi

Mengingat kendala memori, setiap tag hanya dapat menyimpan beberapa token. Pengisian token yang sering menyebabkan ketidaknyamanan yang tidak dapat diterima dalam praktik. Oleh karena itu, kami ingin menemukan cara untuk mengaktifkan pembuatan token dinamis dari beberapa token yang telah diinstal sebelumnya. Selain itu, waktu bagi pembaca untuk mencari token tertentu adalah O(n) dalam desain awal. Kami ingin mengurangi overhead ini menjadi O(1). Lebih penting lagi, kami berharap semua keuntungan dari desain awal, termasuk tidak ada persyaratan fungsi hash kriptografi, overhead komputasi yang rendah untuk tag, dan overhead komunikasi yang rendah untuk pembaca dan tag, dapat dipertahankan dalam desain baru kami.

3.4.2 Ikhtisar

Biarkan tag arbitrer t dalam sistem sudah diinstal sebelumnya dengan token dasar u, dilambangkan dengan [bt^1, bt^2, … , bt^u], masing-masing panjangnya sedikit. Token dasar ini dapat digunakan untuk memperoleh token dinamis untuk otentikasi. Selain itu, kami memperkenalkan jenis kunci tambahan lain yang disebut indikator dasar untuk menghasilkan indikator yang mendukung derivasi token dinamis. Misalkan t menyimpan v indikator dasar yang dilambangkan dengan [bi^1, bi^2, … , bi^v], masing-masing panjangnya b-bit. Biarkan tk mewakili token a-bit saat ini, dan ic menjadi indikator b-bit saat ini. Semua token dasar, indikator dasar, token, dan indikator juga disimpan di server pusat. Ide kami adalah membiarkan pembaca dan tag secara independen menghasilkan token acak yang sama dengan mengikuti instruksi yang dikodekan dalam indikator. TAP terdiri dari tiga fase yaitu fase inisialisasi, fase otentikasi, dan fase update, yang akan dijabarkan satu per satu.

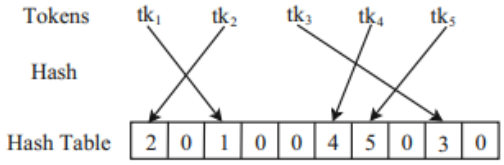
3.4.3 Fase Inisialisasi

Server pusat menyimpan semua kunci tag dalam tabel kunci, dilambangkan dengan KT. Seperti yang ditunjukkan pada Tabel 3.3, setiap entri diindeks oleh indeks tag, yang mendukung akses acak dalam O(1) waktu. Dengan indeks tag idx, kunci t dapat ditemukan di KTŒidx].

Ketika t bergabung dengan sistem, pembaca secara acak menghasilkan array token dasar u [bt^1, bt^2, … , bt^u], array indikator basis v [bi^1, bi^2, ::: , bi^v], token tk dan indikator ic untuk t. Setelah itu, pembaca meminta server pusat untuk menyimpan kunci t tersebut dalam database. Server pusat memasukkan kunci ke entri kosong pertama di KT. Proses pencarian entri kosong dapat dipercepat dengan mempertahankan tabel kecil yang merekam semua entri kosong di KT (mis., karena keberangkatan tag). Jika KT terisi penuh, server pusat menggandakan ukurannya untuk mengakomodasi lebih banyak tag.

Tabel 3.3 Tabel kunci yang disimpan oleh server pusat untuk TAP

Tag index	Tag	Base token array	Token	Base indicator array	Indicator
1	t_1	$[bt_1^1 , \dots , bt_1^u]$	tk_1	$[bt_1^1 , \dots , bt_1^v]$	ic_1
2	t_2	$[bt_2^1 , \dots , bt_2^u]$	tk_2	$[bt_2^1 , \dots , bt_2^v]$	ic_2
⋮	⋮	⋮	⋮	⋮	⋮
n	t_n	$[bt_n^1 , \dots , bt_n^u]$	tk_n	$[bt_n^1 , \dots , bt_n^v]$	ic_n



Gambar 3.3 Tabel hash yang digunakan oleh TAP. Token dari lima tag t_1, t_2, t_3, t_4, t_5 adalah tk_1, tk_2, tk_3, tk_4, tk_5, masing-masing. Setiap token secara acak dipetakan ke slot di tabel hash, di mana indeks tag yang sesuai disimpan.

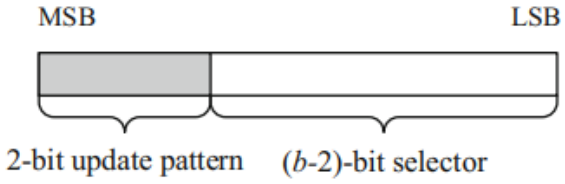
Untuk mengidentifikasi tag berdasarkan tokennya dalam waktu O(1), server pusat memelihara tabel hash HT, memetakan token setiap tag ke indeks tagnya. Biarkan HT terdiri dari l slot. Pada awalnya, setiap slot di HT diinisialisasi ke nol. Setelah t bergabung dengan sistem, pembaca menghitung nilai hash h(tk), di mana fungsi hash h(·) menghasilkan nilai acak dalam [1, l], dan kemudian menempatkan indeks tag idx dari t di slot ke h(tk) dari tabel hash, yaitu, HT[h(tk)] idx (potensi masalah tabrakan hash akan segera diatasi). Gambar 3.3 menyajikan ilustrasi tabel hash yang dibuat untuk token dari lima tag.

3.4.4 Tahap Otentikasi

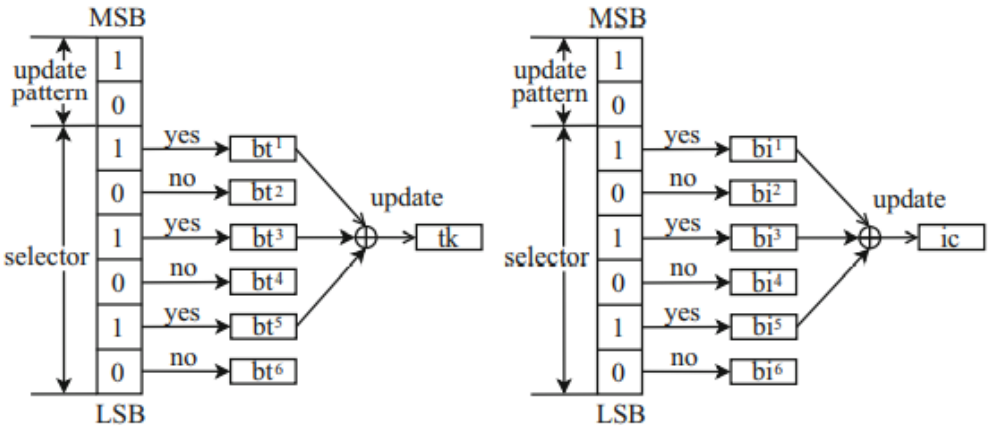
Proses otentikasi TAP mirip dengan desain awal seperti yang ditunjukkan pada Gambar. 3.2. Salah satu perbedaannya adalah pembaca dapat dengan cepat mengidentifikasi tag dari tokennya menggunakan tabel hash. Setelah menerima token tk dari tag t, pembaca terlebih dahulu menghitung h.tk/, kemudian mengakses HT[h(tk)] untuk mengambil indeks tag t, yaitu idx. Jika pembaca menemukan idx 0, itu menegaskan tag itu palsu dan menginformasikan tag kegagalan otentikasi. Jika tidak, pembaca merujuk ke KT[idx] di tabel kunci untuk mengambil token, dan membandingkannya dengan tk token yang diterima. Hanya jika kedua token identik maka tag akan lolos otentikasi. Jika t berhasil diautentikasi, pembaca akan membuat dan mengirimkan token baru untuk mengotentikasi dirinya sendiri. Pembuatan token dengan keacakan yang baik membutuhkan pembaca (lebih tepatnya, server pusat) dan tag untuk memperbarui kunci bersama mereka secara sinkron.

3.4.5 Fase Pembaruan

Untuk menjamin anonimitas, token yang dipertukarkan antara pembaca dan tag harus memiliki keacakan yang baik. Oleh karena itu, pembaca (server pusat) dan tag perlu memperbarui kunci bersama mereka secara sinkron setelah token saat ini digunakan. Kami menekankan bahwa tag akan memperbarui kuncinya setelah menggunakan tokennya saat ini. Oleh karena itu, token yang sama tidak akan pernah digunakan dalam dua otentikasi berturut-turut, yang secara fundamental berbeda dari LAST di mana tag hanya memperbarui pengidentifikasi kuncinya setelah otentikasi berhasil (yang memecah anonimitas).



Gambar 3.4 Struktur indikator b-bit



Gambar 3.5 Plot kiri: Menghasilkan token baru menggunakan token dasar dan pemilih. Plot kanan: Menghasilkan indikator baru menggunakan indikator dasar dan pemilih

Tag t bergantung pada ic indikatornya saat ini untuk memperbarui kuncinya. Gambar 3.4 menunjukkan struktur indikator, yang mencakup dua bagian: (b – 2) bit orde rendah membentuk pemilih, menunjukkan token/indikator dasar mana yang harus digunakan untuk menurunkan token/indikator baru, sedangkan orde tinggi dua bit mengkodekan pola pembaruan. Ketika fase pembaruan dimulai, t menghitung token baru dari token dasar sesuai dengan pemilih. Setiap bit u orde rendah ($u \leq b - 2$) dalam pemilih mengkodekan pilihan token dasar yang sesuai: "0" berarti tidak dipilih, sedangkan "1" berarti dipilih. Untuk semua token dasar yang dipilih, mereka di-XOR untuk menghitung token baru. Karena itu,

$$tk = \bigoplus_{j=1}^{u} ic[j]bt^j, \qquad (3.1)$$

di mana ic [j] adalah bit ke-j dalam ic (asumsikan indeks berbasis satu digunakan) dan $\oplus$ merupakan operator XOR. Plot kiri pada Gambar 3.5 memberikan contoh pembaruan token, di mana bt^1, bt^3, dan bt^5 di antara enam token dasar kebetulan dipilih. Demikian pula, t menurunkan indikator baru dari indikator dasar sebagai berikut:

$$ic = \bigoplus_{j=1}^{v} ic[j]bt^j. \qquad (3.2)$$

Di sisi server, token baru dan indikator baru yang sama dapat dibuat karena berbagi kunci yang sama dengan tag. Selain itu, server juga perlu memperbarui tabel hash. Pertama, server menyetel HT[h(tk)] (token lama) ke 0, dan setelah membuat token baru, server menetapkan HT[h(tk)] = idx.

Setelah memperbarui token dan indikator, server pusat dan tag perlu memperbarui lebih lanjut token dasar dan indikator dasar yang dipilih. Proses pembaruan untuk token dasar atau indikator dasar apa pun yang dipilih mencakup dua langkah: Pergeseran melingkar kiri satu bit, dan pembalikan bit dengan mengikuti pola pembaruan 2 bit tertentu:

1. Pola $(00)_2$: tidak ada flip yang dilakukan;
2. Pola $(01)_2$: membalik bit ke-j jika $j \equiv 0 \pmod 3$;
3. Pola $(10)_2$: membalik bit ke-j jika $j \equiv 1 \pmod 3$;
4. Pola $(11)_2$: membalik bit ke-j jika $j \equiv 2 \pmod 3$.

Jelas, bit ke-i dan ke-j dapat dibalik bersama-sama jika dan hanya jika $j \equiv 0 \pmod 3$. Alasan skema pembaruan ini adalah bahwa jika parameter a dan b diatur dengan benar, dua bit dalam token dasar atau indikator dasar memiliki peluang untuk tidak dibalik, sehingga mengurangi ketergantungan timbal baliknya. Kami akan memberikan bukti formal segera. Kami menekankan bahwa semua kunci hanya disimpan di server pusat daripada setiap pembaca

tunggal. Oleh karena itu, proses pembaruan kunci tag yang dipicu oleh satu pembaca bersifat transparan bagi pembaca lainnya (pembawa tag hanya dapat muncul di satu lokasi pada satu waktu).

3.4.6 Analisis Keacakan

Seperti yang disyaratkan oleh model anonim kami, token yang dihasilkan oleh TAP harus acak dan tidak dapat diprediksi. Keacakan adalah properti probabilistik yang harus dijelaskan dalam hal probabilitas. Kita buktikan terlebih dahulu teorema berikut:

Teorema 1. *Jika $a \geq 2$ dan $a \not\equiv 0 \pmod 3$, harus ada bilangan bulat positif w, di mana $1 \leq w \leq a$, sehingga dua bit yang berbeda dalam satu token dasar akan pindah ke posisi yang tidak dapat dibalik bersama setelah token dasar diperbarui w kali.*

Bukti. Kami melacak dua bit sewenang-wenang dalam token dasar bt^j, dilambangkan dengan variabel acak X dan $Y \in \{0, 1\}$. Misalkan X dan Y awalnya terletak di bit pth dan bit q dari bt^j ($1 \leq p \leq q \leq a$), masing-masing, dan pembaruan w dilakukan ($1 \leq w \leq a$). Dua kemungkinan kasus perlu dipertimbangkan menurut posisi awalnya:

Kasus 1: q—p $6 \not\equiv 0 \pmod 3$ dan a + p—q $\not\equiv 0 \pmod 3$. Pertama, jika $q + w \leq a$, X dan Y masing-masing pindah ke $bt^j[p + w]$ dan $bt^j[q + w]$. Sejak $(q + w) – (p + w) \not\equiv 0 \pmod 3$, tidak bisa dibalik. Kedua, jika $p + w \leq a \leq q + w$, maka X pindah ke $bt^j[p + w]$ dan Y pindah ke $bt^j[q + w - a]$. Karena $(p + w) – (q + w – a) \not\equiv 0 \pmod 3$, mereka masih tidak bisa dibalik. Akhirnya, jika p w > a, X dan Y masing-masing sekarang berada di $bt^j[p + w - a]$ dan $bt^j[q + w - a]$. Demikian pula, karena $(q + w – a) – (p + w – a) \not\equiv 0$ ((mod 3), keduanya tidak dapat dibalik. Oleh karena itu, X dan Y tidak akan pernah dibalik dalam kondisi seperti itu.

Kasus 2: $q – p \equiv 0 \pmod 3$ atau $a + p - q \equiv 0 \pmod 3$ (perhatikan bahwa sama sekali tidak akan $q – p \equiv a + p - q \equiv 0 \pmod 3$ karena $a \not\equiv 0 \pmod 3$). Jika $q – p \equiv 0 \pmod 3$ dan $a - q < w \leq a - p$, X pindah ke $bt^j[p + w]$ dan Y pindah ke $bt^j[q + w - a]$. Karena $(p + u) – (q + u – a) \not\equiv 0 \pmod 3$, mereka pindah ke posisi yang tidak bisa dibalik. Sebaliknya, jika $a + p – q \equiv 0 \pmod 3$, X dan Y tidak akan dibalik bersama di awal, tetapi menjadi mungkin setelah w diperbarui selama $a – q < w \leq a - p$.

Oleh karena itu, X dan Y memiliki peluang untuk tidak dibalikkan bersama dalam pembaruan terlepas dari posisi awal mereka. Sebelum menyelidiki keacakan token turunan, pertama-tama kami mempelajari keacakan token basis arbitrer selama pembaruannya. Kami memiliki lemma berikut:

> **Lemma 1.** *Jika pola pembaruan dalam indikator adalah acak, bit sembarang dalam token dasar menjadi 0 atau 1 dengan probabilitas yang sama menggunakan skema pembaruan kami.*

Bukti. Mari kita lacak satu bit arbitrer dalam bt^j, dilambangkan dengan variabel acak $X \in \{0, 1\}$. Misalkan X saat ini terletak di posisi $bt^j[i]$, di mana $1 \leq i \leq a$. Ketika bt^j diperbarui, X digeser ke

kiri dan kemudian dibalik dengan probabilitas 0,25 jika pola pembaruannya acak. Oleh karena itu, matriks transisi untuk X selama setiap pembaruan adalah $P_1 = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{4} \end{pmatrix}$. Menggunakan dekomposisi nilai singular (SVD),

$P_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix}$. Oleh karena itu, matriks transisi untuk X setelah w diperbarui adalah

$$P_1{}^w = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}^w \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + \frac{1}{2}^{w+1} & \frac{1}{2} - \frac{1}{2}^{w+1} \\ \frac{1}{2} - \frac{1}{2}^{w+1} & \frac{1}{2} + \frac{1}{2}^{w+1} \end{pmatrix},$$

yang konvergen ke $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$. Oleh karena itu, X menjadi 0 atau 1 dengan probabilitas yang sama. Sekarang mari kita selidiki lebih lanjut dua bit arbitrer dalam token dasar, dan kita memiliki lemma berikut:

> ***Lemma 2.*** *Jika pola pembaruan dalam indikator adalah acak, dua bit arbitrer dalam token dasar independen di bawah skema pembaruan kami.*

Bukti. Pertimbangkan dua bit arbitrer, dilambangkan dengan variabel acak X dan Y, dalam token dasar bt^j. Misalkan X dan Y mula-mula terletak pada bit pth dan bit q dari bt^j ($1 \leq p \leq q \leq a$), masing-masing. Matriks transisi ketika X dan Y tidak dapat dibalik dan dapat dibalik adalah

$$P_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{pmatrix}, \text{ and } P_3 = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix},$$

Asumsikan bahwa di antara pembaruan w ($w \geq a$), X dan Y tidak dapat dibolak-balik bersama selama β kali, sedangkan dapat dibolak-balik beberapa γ kali. Kita tahu dari Teorema 1 bahwa $\beta \geq 1$, jadi kita memiliki $\beta \geq 1$, $\gamma \geq 0$, dan $\beta + \gamma = w$.

Kasus 1: $\gamma > 0$. Kami memiliki

$$P_2{}^{\beta} \times P_3{}^{\gamma} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

untuk setiap kombinasi β dan γ.

Kasus 2: = 0. Oleh karena itu, P_2^{β} x P_3^{γ} = P_2^{w} . Menggunakan SVD, kita dapat menghitung setiap entri konvergen ke $\frac{1}{4}$. Oleh karena itu, dua bit arbitrer dalam token dasar berpasangan. Dengan kedua lemma di atas, kita dapat membuktikan teorema berikut mengenai keacakan token yang diturunkan:

$$P_2{}^{w} = \begin{pmatrix} \frac{1}{4} + \frac{1}{2}^{w+1} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} - \frac{1}{2}^{w+1} \\ \frac{1}{4} & \frac{1}{4} + \frac{1}{2}^{w+1} & \frac{1}{4} - \frac{1}{2}^{w+1} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} - \frac{1}{2}^{w+1} & \frac{1}{4} + \frac{1}{2}^{w+1} & \frac{1}{4} \\ \frac{1}{4} - \frac{1}{2}^{w+1} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} + \frac{1}{2}^{w+1} \end{pmatrix},$$

Teorema 2. *Jika indikatornya acak, setiap bit dalam token turunan memiliki probabilitas yang sama untuk menjadi 1 atau 0, dan dua bit arbitrer dalam token turunan adalah independen menggunakan skema pembaruan kami.*

Bukti. Pertimbangkan bit ke-i dari token turunan tk, dilambangkan dengan tk[i] ($1 \leq i \leq a$). Kita tahu tk[i] = $\bigoplus_{j=1}^{u}$ ic[j] bt^{j} [i]. Biarkan N0 menjadi variabel acak dari jumlah token yang bit ke-i adalah 1, tunduk pada $N_0 \geq 0$, $N_1 \geq 0$ dan $N_0 + N_1 = u$. Menurut Lemma 1 dan independensi token dasar yang berbeda, N_0 mengikuti binomial distribusi B(u, 0.5), dan $P(N_0 = x) = \binom{u}{x}$ x $\left(\frac{1}{2}\right)^{u}$, dimana $0 \leq x \leq u$. Menghitung tk[i], kita perlu mempertimbangkan dua kemungkinan kasus:

Kasus 1: $N_0 = u$, yaitu tidak ada 1 dalam bit u tersebut. Dalam hal ini, tk[i] harus 0.

Kasus 2: $0 \leq N0 < u$. Dalam hal ini, tk[i] bisa menjadi 0 atau 1. Jika tk[i] = 0, itu menyiratkan bahwa jumlah token dasar genap yang bit ke-i adalah 1 yang dipilih, dan probabilitas bersyarat adalah

$$P(tk[i] = 0 \mid 0 \leq N_0 < u) = \frac{2^{N_0} \times \sum_{x=0}^{\lceil \frac{N_1}{2} \rceil} \binom{N_1}{2x} - 1}{2^{u} - 1} = \frac{2^{u-1} - 1}{2^{u} - 1}. \quad (3.3)$$

Jadi, peluang untuk tk[i] = 0 adalah

$$
\begin{aligned}
P(tk[i]=0) =& P(N_0 = u) \times P(tk[i]=0 \mid N_0 = u) \\
&+ P(0 \le N_0 < u) \times P(tk[i]=0 \mid 0 \le N_0 < u) \\
=& \frac{1}{2^u} \times 1 + (1 - \frac{1}{2^u}) \times \frac{2^{u-1}-1}{2^u - 1} = \frac{1}{2}.
\end{aligned} \tag{3.4}
$$

Oleh karena itu, P(tk[i] =1) = P(tk[i] =0) = $\frac{1}{2}$. Selain itu, karena tk[i] hanya ditentukan oleh bit ke-i dari token dasar, dan dua bit arbitrer dalam token dasar adalah independen menurut Lemma 2, dua bit arbitrer dalam token turunan juga independen. Analisis keacakan indikator mengikuti jalur yang sama. Hasil simulasi disediakan di bab 3.7 menunjukkan bahwa token dan indikator memiliki keacakan yang sangat baik.

3.4.7 Diskusi

Kebutuhan Memori Untuk mengimplementasikan TAP, setiap tag membutuhkan (u + 1)a + (v + 1)b bit memori untuk menyimpan kunci. Hasil simulasi kami di Sect. 3.7 menunjukkan bahwa a, b, u, dan v dapat ditetapkan sebagai konstanta kecil. Oleh karena itu, kebutuhan memori untuk tag kecil. Kebutuhan memori di server pusat adalah O.n/ untuk menyimpan tabel kunci dan tabel hash.

Overhead Komunikasi Untuk setiap otentikasi, tag hanya perlu mengirimkan satu token a-bit, dan pembaca mengirimkan permintaan otentikasi dan satu respons, keduanya menimbulkan overhead komunikasi O(1).

Overhead Komputasi Online Untuk setiap otentikasi, tag menghasilkan dua token dan melakukan satu perbandingan untuk mengotentikasi pembaca. Semua operasi yang dilakukan oleh tag, termasuk bit-wise XOR, bit flip, dan one-bit left circular shift, sederhana dan efisien perangkat keras. Pembaca (atau server) perlu menghitung dua nilai hash tambahan: satu untuk token yang diterima dari tag untuk mengidentifikasi tag, dan yang lainnya untuk token baru untuk memperbarui tabel hash. Baik tag dan pembaca memiliki overhead komputasi O(1).

3.4.8 Potensi Masalah TAP

TAP memiliki tiga potensi masalah yang harus ditangani.

Serangan Desinkronisasi Pembaca yang tidak sah juga dapat memulai otentikasi dengan mengirimkan permintaan. Tag akan membalas dengan tokennya saat ini, lalu memperbarui kuncinya seperti biasa. Akibatnya, kuncinya berbeda dari apa yang disimpan oleh server pusat. Ketika tag menemukan pembaca yang sah nanti, itu mungkin akan gagal otentikasi karena token saat ini tidak cocok dengan yang disimpan di server pusat.

Replay Attack Saat melakukan serangan desinkronisasi, musuh dapat merekam token yang diterima. Kemudian ia dapat mengirim ulang token untuk mengotentikasi dirinya sendiri. Karena token valid, itu akan melewati otentikasi. Dua masalah di atas juga ada dalam desain awal.

Hash Collision Untuk dua tag dalam sistem, nilai hash dari token mereka saat ini mungkin sama, yang disebut hash collision. Dalam hal ini, indeks tag mereka tidak dapat disimpan di slot tabel hash yang sama. Jika tidak, pembaca tidak dapat secara unik mengidentifikasi tag melalui token yang diterima. Selain itu, karena setiap tag menghasilkan tokennya sendiri, mungkin saja dua tag memiliki token yang sama, yang disebut token collision. Tabrakan token adalah kasus khusus tumbukan hash, dan tumbukan token harus mengarah pada tumbukan hash. Kami menemukan bahwa tabrakan hash, meskipun kemungkinannya rendah, dapat menyebabkan masalah pada semua protokol otentikasi RFID anonim menggunakan *hash kriptografi,* tetapi potensi masalah tidak pernah ditangani dengan hati-hati.

3.5 PROTOKOL OTENTIKASI BERBASIS TOKEN DINAMIS YANG DITINGKATKAN

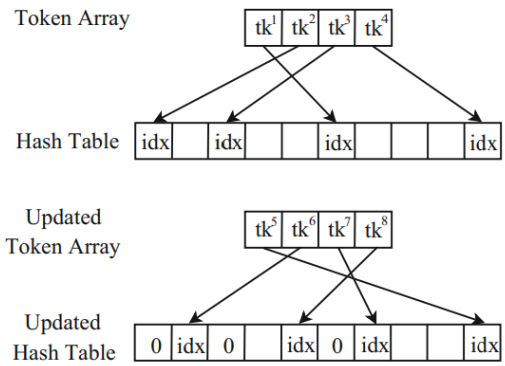
Di bagian ini, kami menyajikan protokol ketiga kami, yang disebut *Enhanced dynamic Token-based Authentication Protocol (ETAP),* untuk mengatasi masalah TAP.

3.5.1 Perlawanan Terhadap Desinkronisasi dan Ulang

Karena serangan desinkronisasi dan serangan replay dapat dilakukan secara bersamaan, kami menanganinya bersama-sama. Tujuan kami ada dua: Pertama, tag yang valid masih dapat berhasil diautentikasi oleh pembaca yang sah setelah beberapa serangan desinkronisasi; Kedua, bahkan jika musuh telah menangkap beberapa token dari tag yang valid, ia tidak dapat menggunakan token tersebut untuk mengotentikasi dirinya sendiri.

Tabel 3.4 Tabel kunci yang disimpan oleh server pusat untuk ETAP

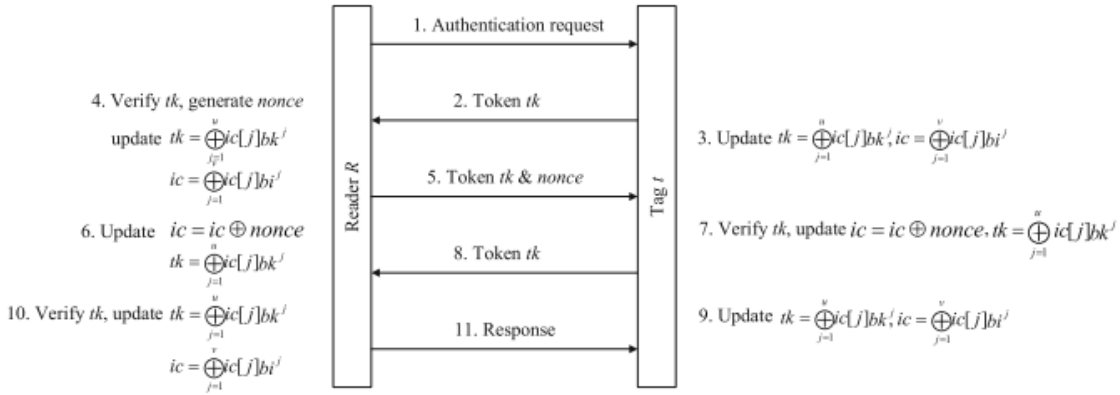
Tag index	Tag	Base token array	Token array	Base indicator array	Indicator
1	t_1	$[bt_1^1, \ldots, bt_1^u]$	$[tk_1^1, tk_1^2, \ldots, tk_1^k]$	$[bt_1^1, \ldots, bt_1^v]$	ic_1
2	t_2	$[bt_2^1, \ldots, bt_2^u]$	$[tk_2^1, tk_2^2, \ldots, tk_2^k]$	$[bt_2^1, \ldots, bt_2^v]$	ic_2
⋮	⋮	⋮	⋮	⋮	⋮
n	t_n	$[bt_n^1, \ldots, bt_n^u]$	$[tk_n^1, tk_n^2, \ldots, tk_n^k]$	$[bt_n^1, \ldots, bt_n^v]$	ic_n



Gambar 3.6 Skema kami melawan serangan desinkronisasi

Untuk membuat protokol kami tahan terhadap serangan desinkronisasi, kami membiarkan server pusat menghitung terlebih dahulu array k token [tk^1, tk^2, ::: , tk^k] dari token dasar, dan token apa pun dapat digunakan untuk mengidentifikasi tag, di mana k adalah parameter sistem yang dapat diatur besar atau kecilnya, tergantung pada memori yang tersedia di server. Pembaca membutuhkan setidaknya satu token untuk mengidentifikasi tag, dan dengan demikian paling banyak k 1 serangan desinkronisasi dapat ditoleransi.[1] Setelah otentikasi timbal balik berhasil, pembaca akan mengisi kembali array token dengan k token baru. Selanjutnya, kami menggunakan proses verifikasi dua langkah untuk menjaga dari serangan replay. Tabel 3.4 menunjukkan tabel kunci yang disimpan oleh server pusat untuk mengimplementasikan ETAP.

Sekarang mari kita uraikan ETAP dengan contoh yang diberikan pada Gambar 3.6. Misalkan k 4 dan pembaca menghitung sebelumnya empat token tk^1, tk^2, tk^3, dan tk^4 untuk tag t dengan indeks tag idx. Selain itu, misalkan token saat ini yang disimpan oleh t adalah tk = tk^2, yang berarti t mungkin berada di bawah satu serangan desinkronisasi dan musuh telah menangkap tk1. Ketika pembaca menerima tk^2 dari t, ia mengakses H[h(tk^2)] untuk mengambil indeks tag idx dari t, dan kemudian array token t dari KT[idx]. Pembaca kemudian melintasi array token sampai menemukan tk2. Setelah itu, pembaca menggunakan token berikutnya dalam array token, tk3 dalam contoh ini, untuk mengotentikasi dirinya sendiri. Jika token yang diterima kebetulan berada di ekor larik, pembaca perlu mendapatkan token baru untuk otentikasi. Untuk mencegah musuh melewati otentikasi menggunakan tk^1,



Gambar 3.7 Mekanisme verifikasi dua langkah ETAP

kami mengadopsi verifikasi dua langkah seperti yang diilustrasikan pada Gambar 3.7. Pada langkah 3, pembaca menyertakan nonce acak b-bit dalam pesannya, dan menantang tag untuk mengirim token lain. Setelah tag mengautentikasi pembaca, ia memperbarui indikatornya

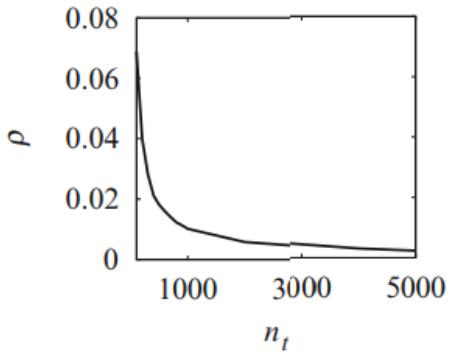
dengan meng-XOR indikator dengan nonce yang diterima (begitu juga pembaca), yang juga berkontribusi pada pengacakan indikator. Setelah itu, tag mendapatkan token baru berdasarkan indikator yang diperbarui, dan mengirimkannya ke pembaca untuk verifikasi kedua. Karena musuh tidak mengetahui token dasar dan indikatornya, ia tidak dapat memperoleh token yang benar untuk lolos verifikasi kedua, membuat serangan replay menjadi tidak mungkin. Setelah otentikasi timbal balik berhasil, pembaca menghasilkan empat token baru untuk mengisi kembali array token. Selain itu, pembaca memperbarui HT dengan mengatur slot yang sesuai dengan token lama ke 0, dan mengatur slot yang sesuai dengan token baru ke idx. Perhatikan bahwa pengisian token dilakukan secara offline oleh server pusat, yang karenanya bukan merupakan masalah kinerja.

3.5.2 Menyelesaikan *Hash Collisions*

Misalkan server pusat menghitung k token untuk setiap tag. Biarkan nt nk menjadi jumlah total token, dan l adalah ukuran tabel hash. Sebuah slot di tabel hash disebut slot kosong, slot tunggal, dan slot tabrakan, masing-masing, jika nol, satu dan beberapa token dipetakan padanya. Ketika setiap token dipetakan ke slot tunggal (tidak ada tabrakan yang terjadi), rasio pemanfaatan p dari tabel hash didefinisikan sebagai

$$\rho = \frac{nk}{l} = \frac{n_t}{l},$$

yang digunakan sebagai metrik kinerja untuk mengevaluasi efisiensi memori.



Gambar 3.8 Rasio pemanfaatan tabel hash ketika fungsi hash tunggal digunakan

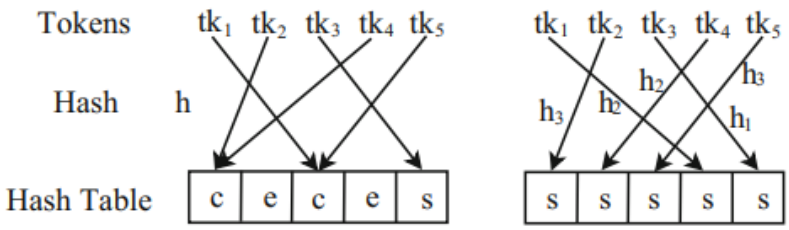
Salah satu pendekatan kandidat untuk mengurangi tumbukan hash adalah dengan menggunakan tabel hash yang besar. Namun, l harus disetel menjadi besar untuk tujuan menghilangkan tabrakan hash secara total, menghasilkan penggunaan memori yang rendah (ρ kecil). Gambar 3.8 menunjukkan rasio pemanfaatan memori dari tabel hash ketika jumlah token yang berbeda perlu disimpan dan satu fungsi hash digunakan. Kita bisa melihat rasio utilisasi p sangat rendah. Selain itu, nilai p turun drastis ketika lebih banyak token dipetakan ke tabel hash. Misalnya, ρ = 0.0023 saat n_t = 5000, artinya lebih dari 99% slot di tabel hash terbuang sia-sia.

Kami mengamati bahwa dua token berbeda yang menyebabkan tabrakan hash di bawah satu fungsi hash mungkin tidak akan memiliki tabrakan di bawah fungsi hash lain. Oleh karena itu, menggunakan beberapa fungsi hash memberikan cara alternatif untuk menyelesaikan tumbukan hash. Ketika fungsi hash tunggal digunakan, probabilitas ps bahwa slot arbitrer adalah slot tunggal adalah

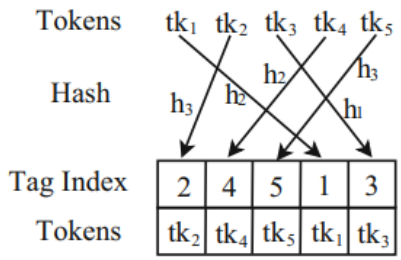
$$p_s = \binom{n_t}{1}\frac{1}{l}\left(1-\frac{1}{l}\right)^{n_t-1} \approx \frac{n_t}{l}e^{-\frac{n_t-1}{l}} \approx \frac{n_t}{l}e^{-\frac{n_t}{l}}. \quad (3.6)$$

Sangat mudah untuk membuktikan bahwa $p_s \leq \frac{1}{e} \approx 0.368$ dan dimaksimalkan ketika l = n_t. Di dalam sebaliknya, jika kita menerapkan dua fungsi hash independen untuk memetakan token ke slot, slot akan memiliki probabilitas hingga 1 – (1 0.368)^2 ≈ 0.601 untuk menjadi singleton di salah satu dari dua pemetaan. Demikian pula, jika kita menerapkan r fungsi hash independen dari token ke slot, probabilitas bahwa slot akan menjadi singleton di salah satu r pemetaan dapat meningkat menjadi 1 – (1 0.368)^r, yang dengan cepat mendekati 1 dengan peningkatan R. Gambar 3.9 menyajikan contoh yang menunjukkan keuntungan menggunakan beberapa fungsi hash dalam mengurangi tumbukan hash. Di plot kiri, hanya satu fungsi hash yang digunakan, dan hanya ada satu slot tunggal, sedangkan di plot kanan, tiga fungsi hash digunakan dan setiap token dipetakan ke slot tunggal.

Terinspirasi oleh pengamatan di atas, ETAP menggunakan r fungsi hash independen, dilambangkan dengan h_1, h_2, … , h_r, untuk menyelesaikan tumbukan hash dan meningkatkan efisiensi memori. Setiap fungsi hash dapat menghasilkan nilai hash yang terdistribusi secara merata pada [1, l]. Untuk menyisipkan token tk dari tag t ke tabel hash, pembaca menghitung h_i(tk) secara berurutan.



Gambar 3.9 Contoh penggunaan beberapa fungsi hash untuk mengurangi tumbukan hash, dimana plot kiri menggunakan satu fungsi hash, dan plot kanan menggunakan tiga fungsi hash. h, h_1, h_2, dan h_3 adalah fungsi hash. "e" berarti slot kosong, "s" berarti slot tunggal, dan "c" berarti slot tabrakan



Gambar 3.10 Contoh mainan tabel hash yang digunakan oleh ETAP. Setiap slot menyimpan indeks hash dan indeks tag.

mulai dari i = 1. Jika slot ke $h_i(tk)$ di tabel hash belum ditempati oleh token apa pun, tk dapat segera ditambahkan ke slot ini. Setiap slot perlu menyimpan nilai token dan indeks tag yang terkait dengan token untuk memudahkan identifikasi token mana yang memang menggunakan slot tertentu. Seperti contoh pada Gambar 3.10, token tk_2 dari t_2 dipetakan oleh fungsi hash h_3 (.) ke slot pertama (slot tunggal) dari HT. Oleh karena itu, HT[0] mencatat indeks tag 2 dan nilai token tk_2. Alih-alih mengimplementasikan r fungsi hash independen, kita dapat menggunakan satu fungsi hash master H dan satu set S seed acak, dan biarkan

$$h_i(tk) = H(tk \oplus S[i]), \qquad (3.7)$$

di mana $\oplus$ adalah operator XOR. Ketika pembaca menerima token tk dari tag untuk otentikasi, itu menghitung hi (tk) ($1 \leq i \leq r$) sampai menemukan bahwa nilai token di slot HT[hi(tk)] cocok dengan tk, di mana ia dapat memperoleh indeks tag yang benar dari itu menandai. Jika tidak ada token yang cocok yang ditemukan, autentikasi tag gagal.

Kami akan segera mengevaluasi efektivitas skema kami untuk menyelesaikan tabrakan hash yang disebabkan oleh token yang berbeda melalui simulasi. Masalah tabrakan token, bagaimanapun, tidak dapat diselesaikan dengan pendekatan ini. Jika dua token identik dikaitkan dengan tag yang sama, itu tidak akan menimbulkan masalah. Tetapi jika mereka terkait dengan tag yang berbeda, pembaca tidak dapat secara unik mengidentifikasi tag dari token yang diterima. Oleh karena itu, token bertabrakan tersebut tidak dapat digunakan untuk otentikasi. Server pusat dapat menyimpan token tersebut dalam CAM (*Content Addressable Memory*) atau *filter Bloom* untuk pencarian cepat. Ketika pembaca menerima token, pertama-tama ia memeriksa apakah itu akan menyebabkan tabrakan token; jika demikian, pembaca perlu meminta token lain dari tag untuk tujuan identifikasi. Kami berharap bahwa peluang untuk tabrakan token kecil selama token yang dihasilkan memiliki keacakan yang baik.

3.5.3 Diskusi

Kebutuhan Memori Persyaratan memori untuk tag untuk mengimplementasikan ETAP sama dengan TAP, yaitu, .(u + 1)a + (v + 1)b bit. Kebutuhan memori di server pusat cukup

meningkat karena tabel kunci yang lebih besar dan tabel hash untuk menyimpan beberapa token untuk setiap tag.

Overhead Komunikasi Untuk setiap otentikasi, tag hanya perlu mengirimkan dua token a-bit, dan pembaca perlu mengirim permintaan otentikasi, satu token a-bit, satu nonce b-bit, dan respons, keduanya menimbulkan O(1) biaya komunikasi. Overhead Komputasi Online Untuk setiap otentikasi, tag menghasilkan tiga token dan melakukan satu perbandingan untuk mengotentikasi pembaca. ETAP memerlukan beberapa overhead komputasi tambahan dari pembaca (server). Pertama, pembaca harus memeriksa apakah token yang diterima adalah token yang bertabrakan, yang memerlukan komputasi O(1). Selain itu, pembaca perlu menghitung paling banyak r nilai hash untuk mengidentifikasi tag, dan melakukan paling banyak k perbandingan untuk menemukan token yang diterima dalam array token. Karena r dan k adalah konstanta kecil, overhead komputasi online untuk pembaca masih O(1).

Biaya Perangkat Keras Perangkat keras untuk tag RFID untuk mengimplementasikan ETAP terdiri dari register geser melingkar, dua register untuk menyimpan hasil antara, beberapa gerbang XOR, dan beberapa RAM untuk menyimpan token dasar, indikator basis v, satu token, dan satu indikator. Kami memperkirakan biaya perangkat keras ETAP mengikuti perkiraan biaya perangkat keras kriptografi tipikal [4, 16] yang tercantum dalam Tabel 3.5. Register geser melingkar adalah sekelompok sandal jepit yang terhubung dalam rantai, yang membutuhkan 12 maks (a, b) gerbang logika. Demikian pula, dua register untuk hasil antara membutuhkan 2 x 12 maks (a, b) gerbang logika. Selain itu, dibutuhkan 2.5 x maks(a, b) gerbang logika untuk mengimplementasikan gerbang XOR. Terakhir, biaya RAM untuk menyimpan token dasar, indikator dasar, token, dan indikator adalah sekitar, $\frac{(u+1)a}{8}$ x 12 + $\frac{(u+1)\,b}{8}$ x 12 gerbang logika. Oleh karena itu, jumlah gerbang logika yang diperlukan untuk mengimplementasikan ETAP adalah kira-kira 38.5 maks(a, b) + 1.5 x (u + 1)a + 1.5 x (v + 1)b. Misalnya, jika kita menetapkan a = b = 16, u = 10, dan v = 6 (alasan pengaturan ini akan segera dijelaskan), ETAP hanya membutuhkan sekitar 1K gerbang logika.

Tabel 3.5 Perkiraan biaya kriptografi tipikal

Blok fungsional	Biaya (gerbang logika)
2 masukan gerbang NAND	1
2 masukan gerbang XOR	2.5
2 masukan DAN gerbang	2.5
FF (Flip-flop)	12
n-byte RAM	*n* x 12

3.6 ANALISIS KEAMANAN

ETAP dirancang agar tahan terhadap serangan desinkronisasi dan serangan replay. Di bagian ini, kami menganalisis lebih lanjut keamanan ETAP di bawah serangan pasif dan aktif. Serangan Token yang Diketahui Dalam ETAP, token ditransmisikan tanpa perlindungan apa pun, yang dapat menyebabkan celah keamanan potensial. Musuh dapat menangkap semua token yang dipertukarkan antara pembaca dan tag, dan menggunakannya untuk menyimpulkan token dasar. Namun, kami memiliki teorema berikut:

Teorema 3. *Memecahkan token dasar dari token yang ditangkap tidak dapat diselesaikan secara komputasi jika jumlah token dasar yang digunakan cukup.*

Bukti. setiap token yang ditangkap memberikan persamaan token dasar. Karena ada u token dasar, setidaknya u persamaan independen diperlukan untuk mendapatkan solusi dari token dasar. Namun, musuh tidak memiliki petunjuk tentang nilai indikator saat ini, yang dapat memiliki keacakan yang sangat baik seperti yang ditunjukkan pada bab 3.7. Oleh karena itu, musuh tidak memiliki cara yang lebih baik daripada mencoba setiap kemungkinan nilai indikator dengan kekerasan. Oleh karena itu, bit u dalam pemilih dan pola pembaruan 2-bit memberikan instantiasi 2^{u+2} untuk setiap persamaan. Oleh karena itu, musuh harus menyelesaikan $(2^{u+2})^u = 2^{u(u+2)}$ set persamaan yang berbeda. Untuk setiap solusi kandidat, musuh mendapatkan token lain, dan membandingkannya dengan yang diambil untuk memverifikasi apakah solusinya benar, yang memerlukan uji coba 2^u lainnya. Akibatnya, total biaya komputasi bagi musuh untuk memecahkan token dasar adalah $2^u \times 2^{u(u+2)} = 2^{u(u+3)}$, yang secara komputasi tidak dapat diselesaikan jika u disetel cukup besar, mis., u = 10.

Anonimitas Karena keacakan token (diverifikasi dalam Bagian 3.7), musuh tidak dapat mengaitkan token apa pun dengan tag tertentu. Menurut Teorema 2, probabilitas bahwa musuh dapat berhasil menebak bit z dari token tag berikutnya berdasarkan token sebelumnya adalah

$$Prob(z\prime = z) \leq \frac{1}{2} + \frac{1}{ploy(s)}, \qquad (3.8)$$

di mana z adalah tebakan musuh dari z, dan poly.s/ adalah polinomial arbitrer dengan parameter keamanan s. Oleh karena itu, musuh tidak memiliki keunggulan yang tidak dapat diabaikan dalam tebakan z, dan ETAP dapat mempertahankan anonimitas tag.

Mengompromikan Resistensi Dalam ETAP, kunci dari setiap tag diinisialisasi dan diperbarui secara independen. Bahkan jika semua tag kecuali dua dikompromikan oleh musuh, itu masih tidak dapat menyimpulkan kunci dari dua tag yang tersisa atau membedakannya berdasarkan token yang ditransmisikan. Oleh karena itu, ETAP kuat terhadap serangan kompromi.

Kerahasiaan Penerusan Kerahasiaan penerusan mensyaratkan bahwa musuh tidak dapat memecahkan pesan sebelumnya yang dikirim oleh tag bahkan jika musuh memperoleh kunci tag saat ini. ETAP memiliki kerahasiaan ke depan yang sempurna karena pada langkah 7 setiap autentikasi, tag akan meng-XOR indikatornya saat ini dengan nonce acak. Bahkan jika musuh

memperoleh semua kunci tag saat ini, ia tidak mengetahui nilai indikator sebelumnya tanpa menangkap semua nonce acak. Oleh karena itu, musuh tidak dapat melakukan operasi kebalikan dari proses pembaruan untuk menghitung token sebelumnya.

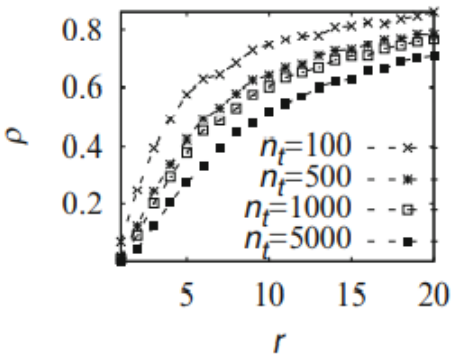
3.7 HASIL NUMERIK

Di bagian ini, pertama-tama kami mengevaluasi efektivitas skema multi-hash dalam menyelesaikan tabrakan hash dan juga meningkatkan efisiensi memori. Setelah itu, kami menjalankan tes keacakan pada token yang dihasilkan oleh ETAP.

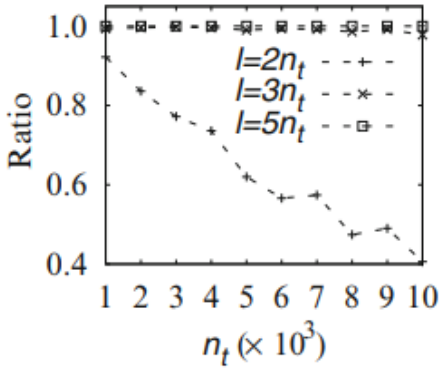
3.7.1 Efektivitas Skema Multi-Hash

Pada set simulasi pertama, jumlah n_t token diatur masing-masing menjadi 100, 500, 1000, dan 5000. Kami memvariasikan jumlah r fungsi hash dari 1 hingga 20. Di bawah setiap pengaturan parameter, kami mengulangi simulasi 500 kali dan mendapatkan nilai rata-rata rasio pemanfaatan. Hasil pada Gambar 3.11 menunjukkan bahwa ρ meningkat secara signifikan dengan peningkatan r pada awalnya, dan secara bertahap mendatar ketika r cukup besar. Pertimbangkan kasus n_t = 5000. Ketika r = 1, kurang dari 0,3% slot yang digunakan. Sebaliknya, ketika r = 10, lebih dari 50% slot terisi. Selain itu, kami mengamati bahwa untuk nt yang lebih besar, ρ yang sesuai sedikit lebih kecil ketika jumlah fungsi hash yang sama digunakan.

Selanjutnya, kami menyelidiki efektivitas pendekatan multi-hash dalam menyelesaikan tabrakan hash. Kami memperbaiki jumlah r fungsi hash menjadi 10, dan memvariasikan jumlah nt token dari 1000 hingga 10.000 pada langkah 1000. Jumlah l slot di tabel hash diatur ke 2 , 3 , dan 5 jumlah token, masing-masing. Di bawah setiap pengaturan parameter, kami menjalankan 500 pengujian dan menghitung rasio pengujian agar tidak terjadi tabrakan hash. Hasil pada Gambar 3.12 menunjukkan bahwa ketika l 2nt, tumbukan hash dapat terjadi dengan probabilitas tinggi, terutama untuk nt yang besar; ketika l ditingkatkan menjadi 5nt, tidak ada lagi hash *collision.*



Gambar 3.11 Rasio pemanfaatan memori dari waktu hash ketika beberapa fungsi hash digunakan



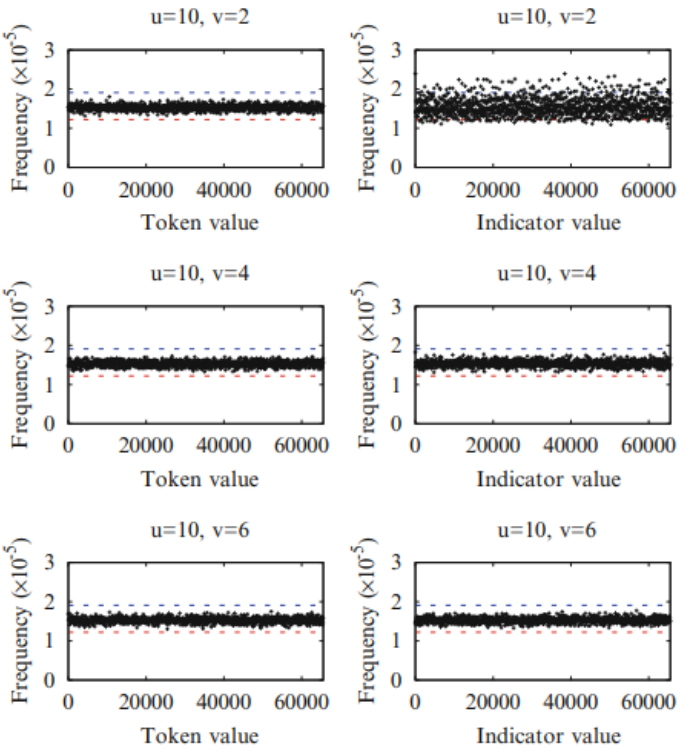
Gambar 3.12 Rasio pengujian yang tidak memiliki tumbukan hash ketika 10 fungsi hash digunakan (r = 10)

3.7.2 Keacakan Tingkat Token

Efektivitas ETAP bergantung pada keacakan token dan indikator. Persyaratan intuitif keacakan adalah bahwa setiap token (indikator) harus memiliki probabilitas yang kira-kira sama untuk muncul. Standar EPC C1G2 menetapkan bahwa untuk generator pseudo-acak 16-bit, probabilitas setiap RN16 16-bit dengan nilai x harus dibatasi oleh $\frac{0.8}{2^{16}} < P(RN16 = x) < \frac{1.25}{2^{16}}$. Untuk mengevaluasi keacakan dari token dan indikator yang dihasilkan oleh ETAP, kami menetapkan a = b = 16, masing-masing, menghasilkan 2^{16} x 500 token dan indikator, dan menghitung frekuensi setiap token atau indikator. Perhatikan bahwa kita dapat menggabungkan beberapa token untuk membentuk token yang lebih panjang jika perlu. Selain itu, kami menetapkan u 10 seperti yang disarankan oleh Teorema 3, dan memvariasikan v = 2, 4, 6 untuk menyelidiki dampaknya terhadap keacakan. Gambar 3.13 menyajikan hasil, di mana garis horizontal putus-putus mewakili batas yang ditentukan oleh EPC C1G2. Kita dapat melihat bahwa indikator memiliki keacakan yang lebih baik dengan peningkatan v, sedangkan keacakan token tidak sensitif terhadap nilai v karena u sudah ditetapkan cukup besar. Selain itu, ketika u = 10 dan v = 4, hanya membutuhkan memori tag 256-bit, baik token maupun indikator memenuhi persyaratan keacakan.

3.7.3 Keacakan Tingkat Bit

National Institute of Standards and Technology (NIST) menyediakan rangkaian statistik untuk uji keacakan, termasuk uji frekuensi monobit, uji frekuensi blok, uji penjumlahan kumulatif, uji lari, uji lari terpanjang dalam satu blok, matriks biner tes peringkat, dll. Karena keterbatasan tempat, kami tidak dapat menjelaskan setiap tes di sini. Diberikan urutan ns bit, ini diterima sebagai acak hanya jika nilai-p yang diamati tidak kurang dari tingkat signifikansi yang ditentukan sebelumnya berdasarkan hipotesis nol H_0.



Gambar 3.13 Pengujian frekuensi untuk token dan indikator yang dihasilkan oleh ETAP, di mana a = b = 16. Setiap titik mewakili token/indikator dan frekuensinya. Dua garis horizontal putus-putus mewakili batas yang diperlukan

Kami menggunakan dua metrik untuk mengevaluasi hasil tes: (1) Proporsi urutan yang lulus tes. Kisaran yang dapat diterima adalah $\hat{p} \pm \sqrt{\frac{\hat{p}(-\hat{p})}{m_2}}$ [17] , di mana $\hat{p}$ = 1 - ∝ dan m_s adalah ukuran sampel; (2) Keseragaman nilai-p yang diamati. Misalkan X adalah variabel acak dengan fungsi kepadatan probabilitas f_X (x), dan Y ∈ [0, 1] menjadi nilai-p dari X. Karena fungsi distribusi kumulatif F(X) dari X meningkat secara monoton, kita memiliki

$$P(Y \le y) = P(\int_X^{\infty} f_X(x)dx \le y) = P(1 - F(X) \le y) \\ = 1 - P(X \le F^{-1}(1-y)) = 1 - (1-y) = y. \quad (3.9)$$

Oleh karena itu, Y ∼ U (0, 1). Kami membagi (0, 1) menjadi sepuluh subinterval yang sama panjang, dan menyatakan jumlah nilai-p di setiap subinterval sebagai F_1, F_2, … , F_{10}, masing-masing. Kita punya

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - \frac{m_s}{10})^2}{\frac{9m_s}{100}} \sim \chi^2(9). \quad (3.10)$$

Buktinya diberikan di bawah ini:

Bukti. Pertimbangkan nilai Fi, di mana 1 i 10. Misalkan Z_{ij} adalah kejadian bahwa p—nilai dari deret ke-j ($1 \leq j \leq m_s$), dilambangkan dengan Y_j, termasuk ke dalam urutan ke-i subinterval $\left(\frac{i-1}{10}, \frac{i}{10}\right)$. Selain itu, biarkan $1_{Z}ij$ menjadi indikator yang sesuai secara acak variabel, yaitu

$$1_{Z_{ij}} = \begin{cases} 1, & \text{if } Y_j \in [\frac{i-1}{10}, \frac{i}{10}), \\ 0, & \text{otherwise.} \end{cases}$$

Oleh karena itu, kami memiliki $F_i = \sum_{j=1}^{ms} 1_{zij}$. Sejak $Y_j \sim U(0,1)$, kami memiliki $E(1_{zij}) = \frac{1}{10}$ dan Var $(1_{zij}) = \frac{1}{10}$ x (1 - $\frac{1}{10}$) = $\frac{9}{100}$. Oleh karena itu, $E(F_i) = \frac{ms}{10}$ dan Var $(F_i) = \frac{9ms}{100}$. Berdasarkan Teorema Limit Pusat (CLT), $\frac{Fi}{ms}$ konvergen ke Norma ($\frac{1}{10}$, $\frac{9}{100ms}$) tanpa gejala. Karena itu, $\left(\frac{\frac{Fi}{ms} - \frac{1}{10}}{\frac{3}{10\sqrt{ms}}}\right)^2 \sim X^2$ (1), dan $X^2 = \sum_{i=1}^{10} \left(\frac{\frac{Fi}{ms} - \frac{1}{10}}{\frac{3}{10\sqrt{ms}}}\right)^2 = \sum_{i=1}^{10} \frac{(Fi - \frac{ms}{10}}{\frac{9ms}{100}} \sim X^2$ (9).

Oleh karena itu, kita dapat menggunakan X^2 tes. Jika statistik yang diamati dari X^2 adalah X^2(obs), nilai p adalah

$$\begin{aligned} P(\chi^2 \geq \chi^2_{obs}) &= \frac{\int_{\chi^2(obs)}^{\infty} e^{-x/2} x^{9/2-1} dx}{\Gamma(9/2) 2^{9/2}} \\ &= \frac{\int_{\chi^2(obs)/2}^{\infty} e^{-x} x^{9/2-1} dx}{\Gamma(9/2)} \\ &= igamc(\frac{9}{2}, \frac{\chi^2(obs)}{2}), \end{aligned}$$

dimana igamc $(c, z) = 1 - \frac{\int_{-\infty}^{z} e^{-x} x^{c-1} dx}{\Gamma(c)}$. Keseragaman dapat diterima jika igam. $\left(\frac{9}{2}, \frac{X^2\ (obs)}{2}\right) \geq 0.0001$ [17]. Kami menetapkan a = b = 16, u = 10, dan v = 4, dan mengubah token yang dihasilkan oleh ETAP menjadi urutan bit. Kami memvariasikan n_s dari 1000, 5000, 10.000 hingga 50.000. NIST menyarankan bahwa $\alpha \geq 0.001$, jadi kami menetapkan $\alpha = 0.01$. Selain itu, kami menetapkan $m_s = 500$, dalam urutan besarnya yang sama dengan $\alpha \geq^{-1}$. Ukuran blok M harus dipilih sedemikian rupa sehingga $M \geq 20$, $M > 0.01n_s$, dan $N_B < 100$, di mana N_B adalah jumlah blok. Kami mengatur $M = 0.02n_s$, jadi $N_B = \frac{ns}{M} = 50$.

Hasil pengujian ditunjukkan pada Tabel 3.6. Kita dapat melihat bahwa urutan bit yang dihasilkan oleh ETAP dapat lulus tes keacakan di bawah semua pengaturan parameter, yang sekali lagi memverifikasi bahwa protokol kami dapat menghasilkan token dengan keacakan yang baik.

Tabel 3.6 Ukuran sampel m_s adalah 500, dan interval kepercayaan yang dapat diterima dari proporsi keberhasilan adalah [0.97665, 1]

Panjang Tes	1000		5000		10,000		50,000	
	Prop.	*p*-value	Prop.	*p*-value	Prop.	*p*-value	Prop.	*p*-value
Frekuensi monobit	0.9920	0.002927	0.9880	0.861264	0.9920	0.719747	0.9900	0.957612
Blok frekuensi	0.9960	0.037076	0.9880	0.538182	0.9880	0.162606	0.9940	0.055361
Jumlah kumulatif (Mode 0)	0.9920	0.119508	0.9860	0.123755	0.9880	0.632955	0.9880	0.986227
Jumlah kumulatif (Mode 1)	0.9920	0.798139	0.9920	0.823725	0.9900	0.877083	0.9900	0.081510
Berlari	0.9940	0.286836	0.9820	0.482707	0.9880	0.146982	0.9860	0.068571
Lari terpanjang	0.9960	0.583145	0.9880	0.554420	0.9860	0.851383	0.9900	0.889118
Peringkat matriks	NA	NA	NA	NA	NA	NA	0.9880	0.004697

Uji peringkat [a]Matrix mensyaratkan bahwa urutan bit terdiri dari setidaknya 38,912 bit. Oleh karena itu, tidak ada pengujian yang dilakukan ketika n_s < 38,912, yang ditandai sebagai NA

3.8 RINGKASAN

Bab ini mencakup otentikasi anonim ringan dalam sistem RFID. Untuk memenuhi batasan perangkat keras dari tag berbiaya rendah, kami mengabaikan hash kriptografi yang intensif perangkat keras dan mengikuti prinsip desain asimetri. Protokol ETAP kami menggunakan teknik baru untuk menghasilkan token acak sesuai permintaan untuk otentikasi anonim. Analisis dan pengujian keacakan menunjukkan bahwa ETAP dapat menghasilkan token dengan keacakan yang sangat baik. Selain itu, ETAP mengurangi overhead komunikasi dan overhead komputasi online ke O(1) per otentikasi untuk tag dan pembaca, yang lebih baik dibandingkan dengan prior art.

BAB 4
MENGIDENTIFIKASI TAG JARINGAN BEBAS NEGARA

Teknologi RFID tradisional memungkinkan tag untuk berkomunikasi dengan pembaca tetapi tidak di antara mereka sendiri. Dengan mengaktifkan komunikasi rekan antara tag terdekat, tag jaringan yang muncul mewakili peningkatan signifikan pada tag RFID saat ini. Mereka mendukung aplikasi dalam skenario yang sebelumnya tidak layak di mana pembaca tidak dapat mencakup semua tag karena keterbatasan biaya atau fisik. Bab ini memperkenalkan masalah mendasar dalam mengidentifikasi tag jaringan. Untuk memperpanjang masa pakai tag jaringan dan membuat protokol identifikasi dapat diskalakan untuk sistem besar, efisiensi energi dan efisiensi waktu adalah yang paling penting. Kami mengungkapkan bahwa desain protokol berbasis pertentangan tradisional akan menimbulkan terlalu banyak energi overhead dalam sistem tag multihop, sementara desain terkoordinasi pembaca yang secara signifikan membuat serial transmisi tag berkinerja jauh lebih baik. Selain itu, kami menunjukkan bahwa penyeimbangan beban penting dalam mengurangi biaya energi terburuk pada tag, dan kami menyajikan solusi berdasarkan nomor seri.

Sisa bab ini disusun sebagai berikut. Bagian 4.1 menyajikan model sistem dan pernyataan masalah. Bagian 4.2 membahas pekerjaan terkait. Bagian 4.3 menjelaskan protokol pengumpulan ID berbasis pertentangan. Bagian 4.4 memperkenalkan protokol pengumpulan ID serial. Bagian 4.5 menyajikan dua teknik untuk meningkatkan efisiensi waktu pengumpulan ID serial. Bagian 4.6 mengevaluasi kinerja protokol kami dengan simulasi. Bagian 4.7 memberikan ringkasan.

4.1 MODEL SISTEM DAN PERNYATAAN MASALAH

4.1.1 Sistem Tag Jaringan

Sistem tag jaringan terdiri dari pembaca dan sejumlah besar objek, yang masing-masing dilampirkan dengan tag. Kami akan menggunakan tag, node, dan tag jaringan secara bergantian di sekuelnya. Setiap tag memiliki ID unik yang mengidentifikasi objek yang dilampirkannya. Pembaca juga memiliki ID unik yang membedakan dirinya dari tag.

Sistem tag jaringan berbeda dari sistem RFID tradisional dengan perubahan mendasar: Tag yang berdekatan dapat berkomunikasi secara langsung. Kemampuan ini memungkinkan jaringan multihop dibentuk di antara tag. Dikembangkan di Universitas Columbia baru-baru ini, prototipe tag jaringan dapat berkomunikasi menggunakan varian CSMA dan slotted ALOHA. Jangkauan transmisi komunikasi antar tag biasanya pendek, sekitar 1–10 m. Tetapi pembaca adalah perangkat yang lebih kuat, dan jangkauan transmisinya bisa jauh lebih besar. Tag yang dapat melakukan komunikasi dua arah langsung dengan sebuah node membentuk lingkungan dari node tersebut.

Tag jaringan diharapkan membawa energi internal yang cukup untuk operasi jangka panjang atau memiliki kemampuan untuk mengumpulkan energi dari lingkungan tempat tag tersebut digunakan. Tag dengan permintaan energi tertinggi terletak di lingkungan pembaca (yaitu, area cakupan) karena mereka harus menyampaikan informasi dari semua tag lain saat data bertemu ke pembaca. Untungnya, tag ini dapat ditenagai oleh gelombang radio pembaca, mirip dengan apa yang dilakukan tag RFID pasif saat ini; pasokan energi mereka terjamin. Sebaliknya, tag yang berada di luar jangkauan pembaca perlu menggunakan energinya sendiri. Pengoperasian tag ini harus dibuat hemat energi.

Pembaca dan tag dalam sistem membentuk jaringan yang terhubung. Dengan kata lain, terdapat setidaknya satu jalur antara pembaca dan tag apa pun sehingga mereka dapat berkomunikasi dengan mentransmisikan data di sepanjang jalur itu. Tag yang tidak dapat dijangkau dari pembaca tidak dianggap berada dalam sistem.

4.1.2 Pernyataan Masalah

Masalah identifikasi tag adalah bagi pembaca untuk mengumpulkan ID dari semua tag jaringan yang dapat dijangkau oleh pembaca melalui beberapa hop dengan bantuan tag perantara yang menyampaikan ID tag yang tidak berada di area jangkauan langsung pembaca. Tujuan kami adalah mengembangkan protokol identifikasi tag yang efisien dalam hal biaya energi dan waktu eksekusi protokol. Kami akan mempertimbangkan biaya energi rata-rata per tag dan biaya energi maksimum di antara semua tag dalam sistem. Biaya energi rata-rata adalah pengukuran keseluruhan pengurasan energi di seluruh sistem, dan biaya energi maksimum adalah pengukuran untuk titik panas terburuk yang dapat menyebabkan tag kehabisan daya dan partisi jaringan.

4.1.3 Tag Jaringan Bebas Negara

Ada dua jenis tag jaringan. Tag jaringan stateful mempertahankan status jaringan seperti tetangga dan tabel perutean dan memperbarui informasi agar tetap mutakhir. Tag ini menyerupai node dalam jaringan sensor biasa.

Sebaliknya, untuk tujuan konservasi energi, tag state-free tidak mempertahankan status jaringan apa pun sebelum operasi, yang membuatnya berbeda dari jaringan tradisional, termasuk jaringan sensor—hampir semua literatur tentang jaringan sensor pengumpul data mengasumsikan stateful model, di mana node sensor menyimpan informasi tentang siapa tetangganya dan/atau bagaimana merutekan data dalam jaringan. Kami mempertimbangkan tag jaringan bebas negara, bukan hanya karena ada sedikit pekerjaan sebelumnya pada jenis node jaringan ini, tetapi juga karena lebih masuk akal untuk masalah identifikasi tag: Pertama, membangun tetangga dan kemudian merutekan tabel di seluruh jaringan mahal dan mungkin memerlukan lebih banyak *overhead* daripada identifikasi tag itu sendiri, yang hanya mengharuskan setiap tag mengirimkan satu nomor (ID-nya) ke pembaca. Kedua, memelihara hubungan tetangga dan memperbarui tabel perutean (karena tag dapat berpindah antar

operasi) memerlukan komunikasi jaringan yang sering, yang tidak bermanfaat untuk operasi identifikasi tag yang jarang.

Merupakan tantangan untuk merancang protokol identifikasi untuk tag jaringan bebas negara. Pertama, karena daya adalah sumber daya yang langka untuk tag, protokol harus hemat energi untuk mengurangi risiko kegagalan jaringan yang disebabkan oleh penipisan energi. Kedua, kita juga harus membuat protokol hemat waktu sehingga dapat menskalakan ke sistem tag besar di mana saluran komunikasi bekerja pada tingkat yang sangat rendah untuk konservasi energi. Ketiga, untuk menghilangkan overhead pemeliharaan keadaan dan dengan demikian menghemat energi, tag diasumsikan bebas-negara, yang berarti bahwa mereka tidak tahu siapa tetangga mereka dan tidak ada struktur perutean yang ada bagi mereka untuk mengirim ID ke pembaca. .

4.1.4 Model Sistem

Apa yang membuat tag menarik adalah kesederhanaannya. Tidak ada spesifikasi tentang bagaimana seharusnya tag jaringan masa depan yang sederhana, tetapi aman untuk mengatakan bahwa kami akan selalu memilih desain protokol yang mencapai efisiensi yang sebanding dengan kebutuhan perangkat keras yang lebih sedikit. Secara umum, setiap tag memiliki energi, memori, dan sumber daya komputasi yang sangat terbatas. Dalam bab ini, kami tidak memerlukan tag untuk mengimplementasikan GPS, mekanisme pelokalan apa pun, atau fungsi kompleks lainnya. Kami menganggap tag bebas status, yang tidak menghabiskan energi dalam mempertahankan informasi status apa pun sebelum pengoperasian. Karena setiap tag hanya dilengkapi dengan satu *transceiver,* maka tidak dapat melakukan transmisi dan penerimaan secara bersamaan. Asumsikan bahwa pembaca dan tag tidak dapat menyelesaikan sinyal yang bertabrakan. Oleh karena itu, sebuah node dapat berhasil menerima transmisi hanya jika hanya ada satu tetangga yang melakukan transmisi.

Untuk tag *state-free*, tidak ada mekanisme (seperti pertukaran suar yang sering antar tetangga) yang melacak perubahan topologi jaringan secara real time. Kami berasumsi bahwa tag tidak bergerak selama operasi identifikasi tag. Misalnya, di gudang, identifikasi tag harian dapat dilakukan secara otomatis di jam kerja setelah barang tidak dipindahkan.

Tabel 4.1 Notasi

Simbol	deskripsi
$T/T^0/T_i$	Tag jaringan dalam sistem
N	Jumlah tag jaringan dalam sistem
f	Ukuran bingkai yang digunakan oleh pembaca
s/s^0	Nomor seri
R	Jangkauan transmisi pembaca
r	Rentang transmisi antar-tag
D_i	Tingkat rata-rata anak-anak dari tag tier-i

L_i	Faktor beban rata-rata dari tag tier-i
N_i	Jumlah tag pada tingkat ke-i
p	Kepadatan distribusi tag jaringan
h	Ukuran kerangka waktu
$RQST_1$	Permintaan untuk menginstruksikan tag untuk melaporkan ID mereka
$RQST_2$	Permintaan untuk menginstruksikan tag yang ditunjuk untuk mengumpulkan ID

Pada siang hari antara identifikasi sebelumnya dan identifikasi berikutnya, objek masih dapat bergerak bebas. Dalam hal operasi identifikasi perlu dilakukan pada siang hari, kita perlu merancang protokol yang membutuhkan waktu sesedikit mungkin untuk menghindari gangguan signifikan pada operasi gudang lainnya karena kebutuhan stasioner pada saat identifikasi.

Untuk menghemat energi, tag jaringan kemungkinan dikonfigurasi untuk tidur dan menunggu operasi secara berkala. Setelah bangun, tag akan mendengarkan siaran permintaan dari pembaca ke dalam jaringan, yang membuat tag kembali tidur atau meminta tag untuk berpartisipasi dalam operasi seperti melaporkan ID-nya. Permintaan siaran akan berfungsi untuk menyinkronkan ulang jam tag secara longgar. Pembaca akan mengatur waktu permintaan berikutnya sedikit lebih lambat dari periode waktu habis yang ditetapkan oleh tag untuk mengimbangi penyimpangan jam dan perbedaan jam pada tag karena penundaan siaran. Waktu tidur yang tepat dari tag dan interval antar-permintaan pembaca harus ditetapkan secara empiris berdasarkan kebutuhan aplikasi dan parameter fisik tag. Notasi yang digunakan dalam bab ini diberikan pada Tabel 4.1 untuk referensi cepat.

4.2 PEKERJAAN TERKAIT

Protokol identifikasi tag untuk sistem RFID tradisional dapat secara luas diklasifikasikan menjadi dua kategori: berbasis ALOHA dan berbasis pohon. Untuk menjalankan protokol identifikasi berbasis ALOHA, pembaca pertama-tama menyiarkan kueri, yang diikuti oleh kerangka waktu yang ditempatkan. Setiap tag secara acak memilih slot waktu dalam bingkai untuk melaporkan ID-nya. Tabrakan terjadi jika slot dipilih oleh beberapa tag. Tag yang tidak menerima pengakuan positif dari pembaca akan terus berpartisipasi dalam bingkai berikutnya. Bingkai dinamis slotted ALOHA (DFSA) menyesuaikan ukuran bingkai putaran demi putaran.

Protokol berbasis pohon mengatur semua ID ke dalam pohon awalan ID. Setiap node dalam pohon memiliki dua node anak yang memiliki satu bit tambahan, “0” atau “1”. ID tag adalah daun dari pohon. Pembaca berjalan melalui pohon. Saat mencapai simpul dalam pohon, ia meminta tag dengan awalan yang diwakili oleh simpul tersebut. Ketika beberapa tag cocok dengan awalan, mereka semua akan merespons dan menyebabkan tabrakan. Kemudian

pembaca pindah ke node anak dengan memperluas awalan dengan satu bit lagi. Jika nol atau satu tag merespons (dalam kasus satu tag, pembaca menerima ID), tag akan bergerak ke atas di pohon dan mengikuti cabang berikutnya.

Untuk lebih meningkatkan efisiensi identifikasi, pengkodean jaringan dan teknik pembatalan interferensi digunakan untuk membantu pembaca memulihkan ID dari sinyal bertabrakan.

4.3 PROTOKOL PENGUMPULAN ID BERBASIS PERTIKAIAN UNTUK SISTEM TAG JARINGAN

Kami tidak mengetahui adanya protokol pengumpulan data yang dirancang khusus untuk model bebas negara yang masuk akal dalam domain tag tetapi tidak diadopsi dalam literatur arus utama jaringan sensor atau jenis sistem nirkabel lainnya. Namun, tidaklah sulit untuk merancang protokol pengumpulan ID untuk tag jaringan berdasarkan teknik yang dikenal dalam sistem nirkabel yang ada. Misalnya, di bagian ini, kita akan mengikuti jalur desain yang jelas berdasarkan *broadcast, spanning tree*, dan transmisi berbasis contention. Protokol yang dihasilkan akan digunakan sebagai tolok ukur untuk perbandingan kinerja (karena tidak ada pekerjaan sebelumnya dalam mengidentifikasi tag jaringan). Pada bagian berikutnya kami akan menunjukkan bahwa teknik yang jelas, bagaimanapun, tidak efisien dan pilihan desain lainnya yang kurang jelas dapat menghasilkan kinerja yang jauh lebih baik.

4.3.1 Motivasi

Salah satu pendekatan langsung untuk tag untuk mengirimkan ID mereka ke pembaca adalah melalui banjir: Karena setiap tag menyiarkan ID-nya dan setiap ID lain yang diterimanya untuk pertama kalinya ke lingkungannya, ID pada akhirnya akan mencapai pembaca. Namun, banjir menyebabkan banyak komunikasi di atas kepala. Selain itu, setiap tag harus menyimpan ID yang telah diterima untuk menghindari duplikasi siaran. Karena sifat dari *flooding*, itu berarti bahwa pada akhirnya setiap tag akan menyimpan semua ID dalam sistem, yang membutuhkan terlalu banyak memori.

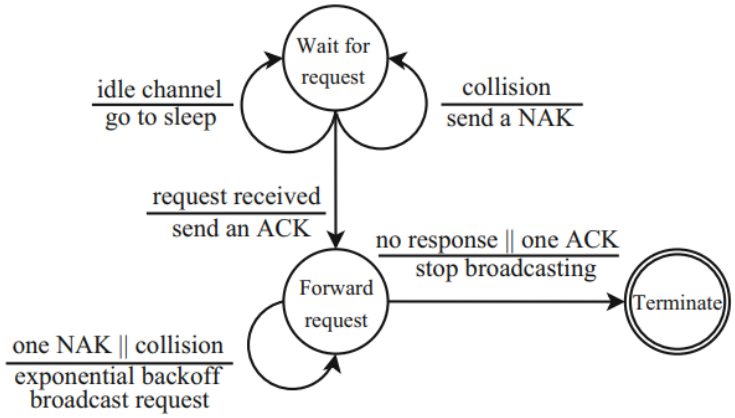
Pendekatan lain adalah meminta tag untuk menemukan tetangganya dan menjalankan protokol perutean untuk membentuk jalur perutean menuju pembaca tepat sebelum mengirim ID (meskipun tag bebas status sebelum operasi). Namun, karena jumlah tetangga bisa mencapai ratusan dalam sistem yang dikemas, *overhead* untuk melakukannya akan tinggi, mengingat hanya satu ID per tag yang akan dikirimkan.

Karena dua pendekatan di atas tidak berfungsi dengan baik, ide kami adalah membuat jalur perutean secara gratis. Agar pembaca dapat memulai proses identifikasi tag, pembaca perlu menyiarkan permintaan ke semua tag. Kita dapat memanfaatkan siaran seluruh jaringan ini secara ekstra untuk mendukung fungsi membangun pohon rentang yang mencakup semua tag, dengan pembaca di akar pohon. Pohon ini akan digunakan untuk mengirimkan ID ke pembaca. Kami menggunakan protokol ALOHA untuk menyelesaikan perselisihan di antara transmisi bersamaan yang dibuat oleh tag jarak dekat.

4.3.2 Meminta Protokol Siaran

Protokol siaran klasik adalah untuk setiap node untuk mengirimkan pesan ketika menerima pesan untuk pertama kalinya. Tetapi menjadi lebih rumit untuk menjamin bahwa semua node menerima pesan: Jika setiap node mengetahui tetangganya, ia dapat terus mentransmisikan pesan sampai menerima pengakuan dari semua tetangga. Namun, lebih hati-hati harus diambil jika node tidak tahu tetangga mereka. Di bawah ini kami menjelaskan secara singkat protokol siaran permintaan (RBP) yang menjamin pengiriman permintaan dari pembaca ke semua tag bebas negara.

Untuk memulai identifikasi tag, pembaca menyiarkan permintaan yang memberi tahu tag untuk melaporkan ID mereka. Permintaan awalnya membawa ID pembaca, yang nantinya akan diganti dengan ID tag saat tag meneruskan permintaan ke orang lain. Diagram transisi status protokol digambarkan pada Gambar 4.1, yang dijelaskan di bawah ini. Status Menunggu Permintaan Setiap tag dimulai dalam status ini dan mengambil tindakan berdasarkan salah satu dari tiga kemungkinan kejadian.



Gambar 4.1 Diagram transisi status protokol RBP.

Setiap lingkaran adalah keadaan, dan setiap panah adalah transisi, di mana peristiwa yang memicu transisi berada di atas garis dan tindakan di bawah garis. Saluran Idle: Saluran tidak aktif, yaitu, tidak ada tetangga yang mentransmisikan apa pun.

1. Permintaan Diterima: Hanya satu tetangga yang meneruskan permintaan, sehingga tag dapat menerima permintaan dengan benar.
2. Collision: Beberapa tetangga meneruskan permintaan, mengakibatkan tabrakan.

Dalam peristiwa (1), tag tidak melakukan apa-apa. Dalam peristiwa (2), tag akan mengakui pengirim bahwa ia telah berhasil menerima permintaan dengan ACK. Sementara itu, ia mengekstrak ID dari permintaan dan menyimpannya sebagai induknya. Setelah itu, pindah ke keadaan *Forwarding Request.* Seperti yang akan kita lihat sebentar lagi, tidak penting apakah ACK diterima dengan benar oleh pengirim permintaan atau tidak. Pengirim akan mengetahui

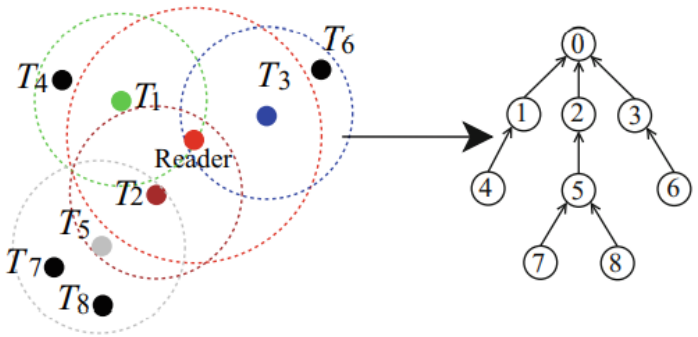
bahwa semua tetangganya telah menerima permintaan ketika tidak mendengar tanggapan apa pun (karena semua tetangga pindah ke status Permintaan Penerusan). Dalam acara (3), tag tidak dapat menyelesaikan bertabrakan. Ini mengirimkan pengakuan negatif (NAK) dan tetap dalam status Menunggu Permintaan.

Status Permintaan Penerusan Untuk memastikan bahwa permintaan akan disebarkan ke seluruh jaringan, setiap tag yang telah menerima permintaan akan terus menyiarkannya dengan backoff eksponensial saat tabrakan sampai semua tetangganya menerima permintaan tersebut. Setiap kali setelah tag menyiarkan permintaan (yang membawa ID tag), ada tiga kemungkinan peristiwa:

1. *No Response*: Tidak ada respon yang diterima dari tetangga manapun.
2. Satu ACK/NAK: Hanya satu respon ACK/NAK yang diterima.
3. Tabrakan: Beberapa respons ACK/NAK dikirim oleh tetangga, yang mengarah ke tabrakan.

Ingat bahwa setiap tetangga dalam keadaan Tunggu Permintaan akan merespon baik ACK atau NAK terlepas dari apakah ia berhasil menerima permintaan atau tidak. Peristiwa (1) harus berarti bahwa semua tetangga telah menerima permintaan dan pindah ke negara bagian lain. Dalam hal ini, tag tidak perlu lagi menyiarkan permintaan. Jika tidak ada respons yang terdengar setelah menyiarkan permintaan untuk pertama kalinya, tag mengetahui bahwa ia tidak memiliki anak dan karena itu merupakan simpul daun di pohon rentang. Pada kejadian (2), jika satu ACK diterima, tag mengetahui bahwa semua tetangganya sekarang telah menerima permintaan tersebut. Oleh karena itu, ia dapat berhenti menyiarkan permintaan tersebut. Jika satu NAK diterima, tag mengetahui bahwa pasti ada tabrakan di tetangga, yang tidak berhasil menerima permintaan. Oleh karena itu, tag harus melakukan backoff eksponensial untuk menghindari tabrakan terus-menerus di saluran. Dalam peristiwa (3), tag tidak dapat menyelesaikan ACK/NAK yang diterima dengan benar dan juga melakukan backoff eksponensial. Sebagai contoh, Gambar 4.2 mengilustrasikan pohon rentang yang dibangun dalam sistem tag jaringan setelah menjalankan RBP, di mana pembaca memiliki ID 0.

Transmisi nirkabel di RBP dapat diimplementasikan baik berdasarkan ALOHA yang tidak memiliki slot atau berdasarkan ALOHA yang memiliki slot. Slotted ALOHA lebih efisien tetapi membutuhkan tag untuk menyinkronkan slotnya. Ketika pembaca mengirimkan permintaannya ke node di sekitarnya, pembukaan transmisi menyediakan sinkronisasi jam dan slot. Demikian pula, ketika tag jauh menerima permintaan untuk pertama kalinya dari tag lain, pembukaan yang terakhir menyinkronkan jam dan slot.



Gambar 4.2 Contoh spanning tree yang dibangun oleh RBP,

dimana ID dari tag Ti adalah i. Setiap lingkaran putus-putus di sebelah kiri memberi tetangga sebuah tag di tengah lingkaran

Teorema 4. Setiap tag akan menerima salinan permintaan yang dikirim oleh pembaca di bawah RBP.

Bukti. Untuk membuktikan dengan kontradiksi, mari kita asumsikan ada tag T yang tidak menerima permintaan setelah menjalankan RBP. Memang benar bahwa tidak ada tetangganya yang menerima permintaan itu. Jika tidak, menurut protokol, tetangga mana pun yang menerima permintaan akan terus menyiarkan permintaan tersebut sampai T menerimanya dan mengakui penerimaannya—setiap kali permintaan ditransmisikan, jika T tidak menerima permintaan dengan sukses, ia akan merespon NACK, menyebabkan pengirim untuk mengirim ulang. Dengan cara yang sama, tetangga dari tetangga T mana pun tidak boleh menerima permintaan tersebut. Menerapkan argumen ini secara rekursif, semua node yang dapat dijangkau dari T tidak boleh menerima permintaan. Dengan asumsi bahwa jaringan terhubung, setidaknya satu tetangga T^0 pembaca dapat dijangkau dari T. Oleh karena itu, T0 tidak boleh menerima permintaan. Ini bertentangan dengan fakta bahwa T^0 terletak di area jangkauan pembaca dan harus menerima permintaan di awal ketika pembaca menyiarkan permintaan untuk pertama kalinya. Oleh karena itu, teorema harus berlaku.

4.3.3 Protokol Pengumpulan ID

Ketika sebuah tag mentransmisikan ID-nya, itu akan menyertakan ID induknya dalam pesan, sehingga simpul induk akan menerimanya sementara tetangga lain akan membuang pesan. Transmisi unicast ini dilakukan berdasarkan ALOHA klasik dengan pengakuan dan backoff eksponensial untuk menyelesaikan tabrakan. Node induk akan meneruskan ID yang diterima ke induknya, dan seterusnya, hingga ID mencapai pembaca.

Eksekusi ICP dilakukan secara paralel dengan RBP: Setelah tag mengetahui ID induknya dari RBP, ia akan mulai mentransmisikan ID-nya ke induk. Saat tag perlu meneruskan ID untuk ICP dan permintaan RBP, kami memprioritaskan penerusan ID karena lebih mudah untuk menyelesaikan unicast.

Teorema 5. Pembaca akan menerima ID dari semua tag dalam sistem setelah eksekusi ICP.
Bukti. Dari Teorema 1, setiap tag dijamin menerima permintaan dan oleh karena itu menemukan induk (dari mana permintaan diterima). Pertimbangkan tag T yang berubah-ubah. Menurut desain ICP, ID T akan dikirim ke induknya sampai diakui secara positif. Induk akan meneruskan ID ke induknya, dan saat proses ini berulang, ID akhirnya akan mencapai pembaca di akar pohon rentang.

4.4 PROTOKOL PENGUMPULAN ID BERSERI

4.4.1 Motivasi

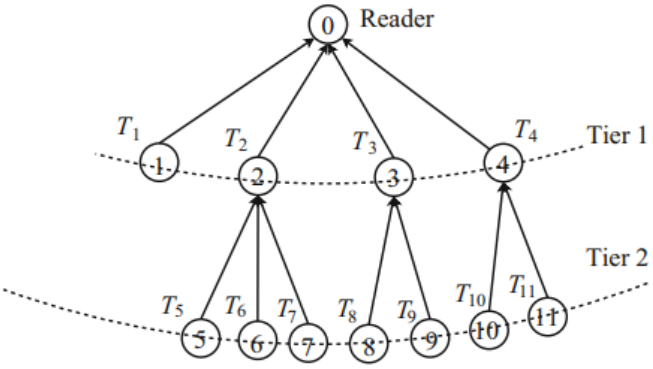
ICP protokol berbasis pertentangan memungkinkan transmisi paralel dengan tag yang tidak mengganggu melalui penggunaan kembali saluran spasial. Dalam kebijaksanaan konvensional, ini adalah keuntungan. Namun, kami menemukan dalam simulasi kami bahwa protokol berbasis pertentangan berkinerja buruk untuk identifikasi tag. Alasannya adalah bahwa meskipun transmisi paralel diaktifkan di antara tag dalam jaringan, pembaca hanya dapat mengambil satu ID pada satu waktu. Pada dasarnya, operasi pengumpulan ID diserialkan pada pembaca, terlepas dari seberapa banyak paralelisme yang dicapai di dalam jaringan tag. Lebih jauh lagi, paralelisme sebenarnya berbahaya karena semakin banyak ID yang dijejalkan ke pembaca secara paralel, semakin banyak pertentangan yang disebabkan oleh pembaca, yang mengakibatkan banyak transmisi yang gagal karena tabrakan, yang berarti biaya energi yang tinggi dan waktu eksekusi protokol yang lama.

Ketika tag dikerahkan secara padat, masalah ini dapat sangat menurunkan kinerja sistem. Dengan pengamatan ini, kami mengambil jalur desain yang berbeda dengan mencoba membuat serialisasi sebagian dari transmisi tag, sehingga hanya sebagian (kecil) dari tag yang akan mencoba untuk mentransmisikan setiap saat. Dengan mengurangi tingkat pertentangan, kami melihat peningkatan kinerja yang drastis. Masalah serius lain dari RBP/ICP adalah bahwa pohon rentang tidak seimbang, menyebabkan pengeluaran energi yang jauh lebih tinggi oleh beberapa tag daripada yang lain. Masalah konsumsi energi yang bias ini dan solusinya akan dijelaskan secara rinci nanti.

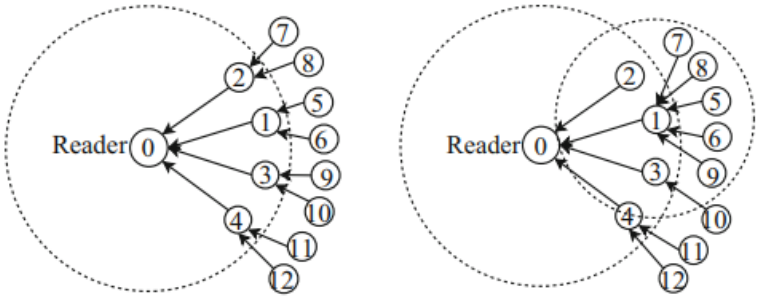
4.4.2 Ikhtisar

Kami memberikan gambaran umum tentang protokol serial kami, SICP. Pembaca memulai dengan mengumpulkan ID di lingkungannya menggunakan ALOHA berbingkai. Contoh ilustratif ditunjukkan pada Gambar. 4.3, di mana pembaca mengumpulkan ID dari tetangga T_1 hingga T_4 (yang membentuk tier 1), sementara semua node lainnya tetap diam. Ketika pembaca menerima ID tag (katakanlah, T_2) bebas dari tabrakan, T_2 harus menjadi satu-satunya tag yang mentransmisikan di seluruh jaringan. Ini juga berarti bahwa tetangga lain dari T_2 dapat mendengar transmisi bebas dari benturan. Node tier-2 ini, T_5 hingga T_7, menetapkan T_2 sebagai induknya.

Setelah mengumpulkan semua tier-1 ID, reader secara berurutan menginformasikan setiap tier-1 node untuk selanjutnya mengumpulkan ID dari anak-anaknya. Misalnya, ketika pembaca menginformasikan T_2 untuk melakukan



Gambar 4.3 Setiap saat, hanya ada satu node yang aktif dalam mengumpulkan ID dari tetangga tingkat bawahnya



Gambar 4.4 Plot kiri: pohon merentang yang kira-kira seimbang; Plot kanan: pohon rentang yang bias, di mana tag 1 mengirimkan ID-nya ke pembaca terlebih dahulu dan sejumlah besar node di lingkungannya memilihnya sebagai induknya, menyebabkan pohon bias. Nomor seri tag ditampilkan di dalam lingkaran. Setiap panah mewakili hubungan anak-orang tua

Jadi, semua node tingkat-1 lainnya akan tetap diam. Saat T_2 mengirimkan permintaan ID, hanya turunannya (T_5, T_6 dan T_7) yang akan merespons. Proses yang sama seperti yang dijelaskan pada paragraf sebelumnya akan berulang; hanya saja kali ini T_2 mengambil peran sebagai pembaca.

Setelah T_2 mengumpulkan ID dari semua anak-anaknya, itu akan meneruskan ID ke pembaca, yang kemudian akan pindah ke node tier-1 berikutnya. Setelah menghabiskan semua node tier-1, ia akan pindah ke node tier-2, satu per satu dan tier demi tier, hingga ID semua node dalam jaringan dikumpulkan.

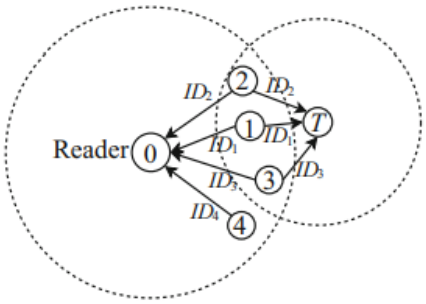
Di bawah ini pertama-tama kami akan memperkenalkan masalah konsumsi energi yang bias, memberikan solusi, dan kemudian menjelaskan serialisasi rekursif.

4.4.3 Konsumsi Energi yang Bias

Ketika sebuah tag mengirimkan ID-nya ke reader, tetangganya di luar jangkauan reader dapat mendengar ID-nya. Mereka dapat menggunakan tag ini sebagai induknya. Seperti yang diilustrasikan di plot kiri Gambar 4.4, kami lebih memilih pohon merentang yang kira-kira seimbang di mana setiap simpul berfungsi sebagai induk untuk jumlah anak yang sama. Namun pada kenyataannya, tag yang memberikan ID-nya kepada pembaca sejak dini akan cenderung memiliki lebih banyak anak. Sebuah contoh diberikan di plot kanan Gambar 4.4. Misalkan tag 1 mengirimkan ID-nya ke reader terlebih dahulu. Mendengar ID-nya, tag 5–9 akan memilih tag 1 sebagai induknya. Ketika tag 2 mengirimkan ID-nya di lain waktu, tidak ada tag yang tersisa untuk memilih tag 2 sebagai induk meskipun tag 7-8 berada dalam kisaran tag 2—ingat bahwa mereka telah memilih tag 1. Dalam hal ini, tag 1 harus meneruskan lebih banyak ID, sehingga menguras energi lebih cepat daripada yang lain. Tingkat keparahan masalah berkembang pesat dengan meningkatnya jumlah tingkatan karena banyak anak-anak tag 1 cenderung memperoleh lebih banyak anak-anak mereka sendiri dan ID tersebut akan melewati tag 1 ke pembaca.

Konsumsi energi yang tidak merata menyebabkan beberapa tag kehabisan energi lebih awal, yang dapat mengakibatkan partisi jaringan. Masalah yang sama juga terjadi untuk RBP/ICP di mana tag yang menerima dan meneruskan permintaan sejak awal selama siaran di seluruh jaringan mungkin berakhir dengan sejumlah besar anak.

Kami mengamati bahwa sebuah tag mungkin mendengar beberapa transmisi ID dari waktu ke waktu dan dengan demikian memiliki banyak kandidat untuk dipilih dari induknya, seperti yang ditunjukkan oleh Gambar 4.5 di mana T dapat memilih induknya dari tiga node tier-1. Idealnya, tag harus memilih induknya secara acak secara acak dari kandidat. Namun, karena tabrakan, setiap kandidat mungkin harus mengirimkan ulang ID-nya beberapa kali sebelum pembaca berhasil menerimanya. Untuk menghindari memberi lebih banyak kesempatan kepada kandidat yang mentransmisikan ulang ID-nya lebih sering, tag mungkin menyimpan ID dari semua kandidat yang dikenal untuk menyaring pendengaran ganda. Namun, kelemahan serius dari pendekatan ini adalah bahwa biaya memori bisa tinggi jika tag memiliki banyak kandidat untuk induknya dalam sistem di mana objek yang ditandai dikemas bersama-sama. Kami ingin menunjukkan bahwa tag tipikal memiliki memori yang sangat terbatas.



Gambar 4.5 Sebuah tag dapat memilih induknya dari beberapa kandidat, di mana panah mewakili transmisi ID (atau siaran)

4.4.4 Nomor Seri

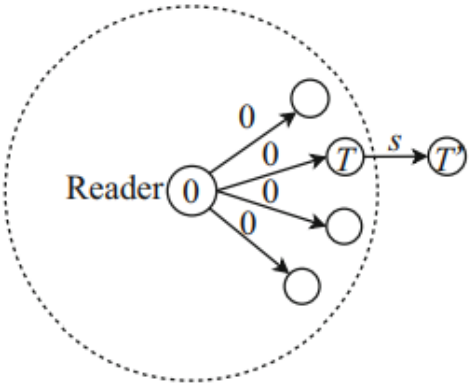
Kami hadirkan solusi untuk konsumsi energi yang bias berdasarkan nomor seri. Dalam protokol kami, setiap tag akan secara dinamis diberi nomor seri dari 1 hingga N, di mana N adalah jumlah tag. Nomor seri pembaca adalah 0.

Pertama-tama mari kita pertimbangkan lingkungan pembaca saja; tingkatan lainnya akan dijelaskan kemudian. Pembaca memulai protokol dengan menyiarkan permintaan pengumpulan ID yang dilambangkan dengan RQST1, membawa nomor seri dan ukuran bingkai f . Permintaan tersebut diikuti oleh kerangka waktu slot f. Setiap tag yang menerima permintaan akan menetapkan nomor seri 0 (yaitu, pembaca) sebagai induknya dan kemudian secara acak memilih slot dalam kerangka waktu. Itu menunggu hingga slot yang dipilih untuk melaporkan ID-nya ke pembaca. Jika hanya satu tag yang memilih slot tertentu, ID-nya akan diterima dengan benar oleh pembaca, yang membalas ACK ke tag di slot yang sama. ACK membawa nomor ID yang telah berhasil diterima pembaca sejauh ini. Nomor ini ditetapkan sebagai nomor seri tag; jumlahnya unik di seluruh sistem karena sifatnya yang meningkat secara monoton. Sebuah tag dapat diidentifikasi baik dengan ID atau nomor seri yang ditetapkan. Setelah menerima ACK, kami memerlukan tag untuk menyiarkan nomor seri yang ditetapkan di lingkungannya. Oleh karena itu, setiap slot waktu berisi transmisi ID, transmisi ACK, dan transmisi nomor seri. Jika transmisi ID bebas benturan, demikian juga dengan dua transmisi lainnya. Meskipun sebuah tag mungkin perlu mentransmisikan ulang ID-nya beberapa kali karena tabrakan, tag akan mengirimkan nomor seri yang ditetapkan satu kali, hanya pada saat ACK diterima.

Jika pembaca mengamati setiap tabrakan dalam kerangka waktu, itu akan menyiarkan permintaan lain dengan kerangka waktu lain untuk mengumpulkan lebih banyak ID. Jika tidak ada tabrakan yang diamati, pembaca telah mengumpulkan semua ID dari lingkungannya dan akan melakukan serialisasi rekursif (akan dibahas) untuk mengumpulkan ID di luar lingkungannya.

4.4.5 Seleksi Induk

Pertimbangkan nilai acak dari T, dilambangkan sebagai T^0, yang belum menetapkan induknya. Seperti yang diilustrasikan pada Gambar 4.6, T^0 tidak boleh berada di lingkungan pembaca karena tag di lingkungan itu menetapkan nomor seri 0 sebagai induknya saat menerima permintaan dari pembaca untuk pertama kalinya. Ketika T^0 menerima nomor seri untuk pertama kalinya, ia akan menetapkan nomor tersebut sebagai induknya, yang dapat berubah ketika T0 menerima lebih banyak nomor seri dari tag lain (calon induk). Ingatlah bahwa setiap tag menyiarkan nomor serinya hanya sekali. Properti ini memungkinkan kami untuk merancang algoritma pemilihan induk (PSA) berikut yang menjamin setiap kandidat memiliki kesempatan yang sama untuk dipilih sebagai induk: Setiap tag mempertahankan dua nilai, induknya dan penghitung c untuk jumlah kandidat yang telah ditemukan sehingga jauh. Penghitung diinisialisasi ke nol. Setiap kali ketika T^0 menerima nomor seri s^0 dari tetangga, itu meningkatkan c satu dan kemudian menggantikan induk saat ini dengan s^0 dengan probabilitas 1. Dengan menggunakan PSA ini, kita memiliki teorema berikut:



Gambar 4.6 T^0 menetapkan T sebagai induknya

Teorema 6. Misalkan sebuah tag memiliki m kandidat untuk parent. Setiap kandidat memiliki probabilitas yang sama yaitu $\frac{1}{m}$ untuk dipilih sebagai induk tag pada akhirnya.

Bukti. Untuk kandidat yang ditemukan ke-j ($1 \leq j \leq m$), ia menjadi parent terakhir hanya jika ia menggantikan parent yang dipilih sebelumnya, dan tidak pernah digantikan oleh kandidat yang ditemukan kemudian. Oleh karena itu, peluang terpilihnya tag sebagai parent pada akhirnya adalah

$$\frac{1}{j}\prod_{l=j+1}^{m}(1-\frac{1}{l})=\frac{1}{m}, \qquad (4.1)$$

menyiratkan bahwa setiap kandidat memiliki kemungkinan yang sama untuk menjadi orang tua. Keuntungan lain menggunakan nomor seri daripada ID untuk identifikasi induk adalah bahwa ID—biasanya 96 bit atau lebih untuk tag RFID—jauh lebih panjang daripada ukuran nomor seri,

$[log_2 N]$, di mana N adalah jumlah maksimum tag di sistem. Misalnya, bahkan jika N = 1.000.000, nomor seri hanya 20 bit.

4.4.6 Serialisasi di Tingkat Dua

Setelah pembaca mengumpulkan semua ID dari lingkungannya, setiap tag di lingkungan tersebut akan mendapatkan nomor seri yang unik. Ingat bahwa tag ini merupakan tingkat pertama dari jaringan. Pembaca kemudian membuat serial proses pengumpulan ID berikutnya dengan mengirimkan nomor seri tag tier-1 satu per satu, untuk memerintahkan tag yang sesuai untuk mengumpulkan ID dari tetangganya, dengan tag tier-1 lainnya tetap menganggur.

Pembaca mulai dengan mengirimkan jenis permintaan lain, dilambangkan dengan RQST2, yang mencakup nomor seri 1 dan nomor s ID yang telah diterima sejauh ini. Sebagai tanggapan, tag dengan nomor seri 1, dilambangkan sebagai T_1, mengirimkan permintaan pengumpulan ID RQST1, membawa nomor serinya sendiri 1 dan ukuran bingkai f . Permintaan tersebut menyebabkan tetangga yang bukan tier-1 menyelesaikan pemilihan induknya; node ini adalah tier-2. Perhatikan bahwa beberapa dari mereka mungkin telah memilih node selain T_1 sebagai orang tuanya. Oleh karena itu, ketika node tier-2 menerima permintaan dari T_1, hanya jika orang tua yang dipilihnya cocok dengan nomor seri dalam permintaan, ia akan mengirimkan ID-nya dalam jangka waktu berikutnya; jika tidak, ia dapat tidur selama slot f. Jika T_1 dengan benar menerima ID dalam slot dari anak T10 , itu meningkatkan nilai s satu per satu dan mengirim kembali ACK dengan s sebagai nomor seri yang ditetapkan ke T_1^0 , yang pada gilirannya menyiarkan nomor seri dan nomor tingkatnya (yaitu, 2) di lingkungannya sedemikian rupa sehingga tetangga di tingkat berikutnya dapat menemukannya sebagai salah satu calon orang tua mereka. Ketika sebuah tag menetapkan (atau yang lebih baru menggantikan) induknya, ia juga menetapkan nomor tingkatnya sebagai nomor tingkat induknya ditambah satu; itu tidak boleh mengganti induknya saat ini dengan induk yang nomor tiernya lebih besar.

Mungkin diperlukan beberapa permintaan T_1 untuk menyelesaikan membaca semua ID dari turunannya. Ini kemudian meneruskan ID ke pembaca. Setelah mengakui T_1, pembaca mengirimkan perintah untuk memicu proses pengumpulan ID pada tag tier-1 berikutnya.

Setelah pembaca menyelesaikan proses ini dengan semua tag tier-1, ia telah mengumpulkan ID dari semua tag tier-2. Pembaca juga memiliki informasi untuk membangun pohon merentang yang mencakup node tier-1 dan tier-2, seperti yang diilustrasikan pada Gambar 4.3 di mana nomor seri yang ditetapkan ditampilkan di dalam lingkaran.

4.4.7 Serialisasi Rekursif

Setelah reader memerintahkan semua tag tier-1 satu per satu untuk mengumpulkan ID dari tag tier-2, ia mengulangi proses serialisasi ini secara rekursif untuk mengumpulkan ID lain tier demi tier. Misalkan pembaca telah mengumpulkan ID dari semua tag di tingkat 1 hingga tingkat i dan rentang nomor seri di tingkat i adalah dari x ke y. Pembaca akan mengirimkan RQST2 ke setiap tag tier-i secara berurutan. Perintah tersebut mencakup rangkaian nomor seri di sepanjang jalur di pohon rentang dari root (dikecualikan) ke tag itu, selain jumlah s ID yang

telah diterima pembaca sejauh ini. Misalnya, untuk tag 7 pada Gambar 4.3, perintah akan membawa dua nomor seri, 2 dan 7. (Perhatikan bahwa karena setiap nomor seri berukuran tetap, tidak ada ambiguitas dalam menafsirkan urutan nomor seri.)

Ketika pembaca menyiarkan perintah di lingkungannya, setiap tag yang menerima perintah akan mengekstrak dan membandingkan nomor seri pertama dengan miliknya. Jika dua nomor seri tidak cocok, perintah akan dibuang. Jika tidak, ia akan memeriksa lebih lanjut apakah ada lebih banyak nomor seri dalam perintah. Jika demikian, ia menyiarkan perintah yang tersisa. Proses ini berulang hingga tag cocok dengan nomor seri terakhir dalam perintah. Tag itu akan melakukan pengumpulan ID dengan cara yang sama seperti yang dijelaskan di bab Sebelumnya. ID yang dikumpulkan akan dikirim melalui rantai induk ke pembaca.

Teorema 7. Pembaca akan menerima ID dari semua tag dalam sistem setelah eksekusi SICP.

Bukti. Dibuktikan dengan kontradiksi, kami menganggap setidaknya satu tag T gagal dalam mengirimkan ID-nya ke pembaca. T tidak boleh memiliki orang tua; kita buktikan lagi dengan kontradiksi: Asumsikan bahwa T memiliki induk T^0. Menurut protokol, agar T^0 dipilih sebagai induk, ia harus pembaca atau simpul yang telah berhasil mengirimkan ID-nya dan selanjutnya menyiarkan nomor seri yang ditetapkan. Oleh karena itu, ia akan menerima perintah dari pembaca untuk mengumpulkan ID dari anak-anaknya. Setelah pembaca mengirim perintah ke T^0, T^0 akan menyiarkan permintaan, bebas dari tabrakan karena serialisasi, ke anak-anak hingga semua ID dikumpulkan—yang terjadi ketika tidak ada tabrakan yang terdeteksi dalam kerangka waktu setelah permintaan. Ketika T^0 menerima ID dari T, jika bukan pembaca, itu akan meneruskan ID ke pembaca di sepanjang jalur yang ID-nya sendiri telah berhasil dikirimkan, bebas dari tabrakan karena serialisasi. Hal ini bertentangan dengan asumsi bahwa T gagal dalam menyampaikan ID-nya kepada pembaca. Oleh karena itu, T tidak memiliki orang tua.

Jika T tidak memiliki parent, semua tetangganya harus gagal dalam mengirimkan ID mereka ke reader karena jika tidak, setiap tetangga yang berhasil akan menyiarkan nomor serinya sesuai dengan protokol, yang akan mengakibatkan T memiliki induk setelah T menerima nomor seri.

Jika semua tetangga T gagal dalam mengirimkan ID mereka ke pembaca, dengan alasan yang sama seperti di atas, semua tetangga mereka juga harus gagal. Dengan menerapkan argumen ini secara rekursif, semua tag dalam jaringan harus gagal dalam mengirimkan ID mereka ke pembaca karena jaringan terhubung, yang setidaknya bertentangan dengan fakta bahwa tetangga langsung pembaca dapat mengirim ID mereka ke pembaca melalui slotted ALOHA protokol yang digunakan SICP. Oleh karena itu, teorema terbukti.

Kode semu SICP untuk tag arbitrer T diberikan dalam Protokol 1.

Protocol 1 Serialized ID collection at tag T

```
1: if T has not reported its ID then
2:     if T receives a serial number from a neighbor then
3:         if T has not determined its parent then
4:             T executes PSA for parent selection;
5:         end if
6:     else if T receives a RQST1 request then
7:         if T has not determined its parent then
8:             T sets the candidate parent to its parent;
9:         end if
10:        if the request is sent from T's parent then
11:            T randomly picks a slot in the following time frame to report its ID;
12:        else
13:            T sleeps during the following time frame;
14:        end if
15:    end if
16: else if T receives a RQST2 request then
17:    if the designated tag in the request is T's descendant then
18:        T forwards the request to the target child;
19:    else if the designated tag in the request is T itself then
20:        T performs ID collection from its children;
21:        T forwards the collected IDs to its parent;
22:    else
23:        T discards the request;
24:    end if
25: else if T receives a forwarding request then
26:    if the request is sent from its child then
27:        T receives and forwards the IDs to its parent;
28:    else
29:        T sleeps when its neighbor is forwarding IDs;
30:    end if
31: end if
```

4.4.8 Ukuran Bingkai

Ketika pembaca atau tag mencoba mengumpulkan ID di lingkungannya, permintaannya membawa ukuran bingkai f . Biarkan n menjadi jumlah tag yang merupakan anak dari pembaca atau tag yang mengirim permintaan. Telah diketahui dengan baik bahwa ukuran bingkai yang optimal harus ditetapkan sebagai n, sehingga kemungkinan setiap slot yang membawa satu ID (tanpa tabrakan) dapat dimaksimalkan. Ini dapat dengan mudah dilihat sebagai berikut: Pertimbangkan slot sewenang-wenang. Probabilitas p bahwa satu dan hanya satu tag memilih slot ini untuk ditransmisikan adalah

$$p = \binom{n}{1}\frac{1}{f}\left(1-\frac{1}{f}\right)^{n-1} \approx \frac{n}{f}e^{-\frac{n-1}{f}} \approx \frac{n}{f}e^{-\frac{n}{f}} \qquad (4.2)$$

Ketika *n* besar. Untuk mencari nilai f yang memaksimalkan p, kita ambil turunan orde pertama dari ruas kanan dan set ke nol. Memecahkan persamaan yang dihasilkan, kita memiliki

$$f = n, \qquad (4.3)$$

yang berarti nilai maksimal p adalah e^{-1}. Dalam permintaan berikutnya, karena semakin banyak ID yang dikumpulkan, semakin sedikit tag yang mentransmisikan ID mereka dan ukuran bingkai harus dikurangi.

Namun, kita tidak tahu n. Ada banyak metode estimasi untuk n, bagaimanapun, ditujukan untuk sistem dengan sejumlah besar tag, dalam puluhan ribu. Diketahui bahwa metode estimasi ini sebenarnya tidak efisien jika diterapkan pada jumlah tag yang relatif kecil seperti beberapa ribu atau kurang; jika jumlah tag sangat kecil, waktu estimasi bisa jauh lebih besar daripada waktu yang dibutuhkan untuk menyelesaikan tugas identifikasi tag itu sendiri. Dalam konteks bab ini, kami berharap jumlah anak pembaca atau tag apa pun relatif kecil. Oleh karena itu, tidak ada gunanya menambahkan overhead komponen terpisah untuk memperkirakan n sebelum pembaca (tag) mulai mengumpulkan ID dari lingkungannya.

Solusi kami adalah memperkirakan nilai n secara iteratif dari frame itu sendiri tanpa menimbulkan overhead tambahan. Awalnya, kami menetapkan f menjadi konstanta kecil λ dalam permintaan pertama. Kami menggandakan nilai f di setiap permintaan berikutnya hingga ada setidaknya satu slot kosong yang tidak dipilih oleh tag. Sejak saat itu, kami akan memperkirakan jumlah n dan mengatur ukuran bingkai sesuai dengan permintaan berikutnya. Tanpa menghilangkan keumuman, misalkan kita ingin menentukan ukuran bingkai untuk permintaan ke-i. Biarkan f_j menjadi ukuran bingkai yang digunakan dalam permintaan ke-j, $1 \leq j < i$. Setelah permintaan ke-j, biarkan c_j, s_j, dan e_j menjadi jumlah slot yang dipilih oleh beberapa tag (tabrakan), tag tunggal, dan tag nol, masing-masing. Biarkan mj menjadi jumlah ID yang dikumpulkan secara berurutan setelah permintaan ke-j. Semua nilai ini diketahui oleh pembaca (tag). Proses tag untuk memilih slot secara acak dalam kerangka waktu dapat dilemparkan ke dalam bin dan masalah bola. Dalam bingkai ke-j, $n - m_{j-1}$ tag (bola) dipetakan ke slot fj (tempat sampah). $n - m_{j-1}$.Banyaknya cara yang berbeda untuk menempatkan $n - m_{j-1}$ bola ke f_j bin adalah $f_j^{\,n - mj-1}$. Banyaknya cara memilih ej bin dari fj bin dan membiarkannya kosong adalah $\binom{f_j}{e_j}$.

Selain itu, banyak cara untuk memilih s bola dari n m bola dan memasukkan masing-masing ke dalam salah satu tempat $f_j - e_j$ yang tersisa adalah $\binom{f_j - e_j}{s_j}\binom{n - m_j - 1}{s_j}(s_j!)$. Akhirnya, sisa $n - m_{j-1}$ bola sj harus dibuang ke tempat sampah c_j yang tersisa, masing-masing berisi setidaknya dua bola (collision slot). Pertama-tama kita memilih bola $2c_j$ dan memasukkan dua bola ke masing-masing kotak cj, yang mencakup kemungkinan $\binom{n - m_j - 1 - s_j}{2c_j}\frac{\binom{2}{c_j}}{2^{c_j}}$ Setelah itu, sisa $(n - m_{j-1} - s_j - 2c_j)$ bola dapat dimasukkan ke dalam salah satu tempat c_j, yang

melibatkan $(\mathrm{n} — \mathrm{mj} — 1 - \mathrm{sj} - 2\mathrm{cj})^{c_j}$ dengan cara yang berbeda. Oleh karena itu, fungsi kemungkinan untuk mengamati nilai-nilai ini adalah

$$L(n) = \prod_{j=1}^{i-1} \frac{\binom{f_j}{e_j}\binom{f_j - e_j}{s_j}\binom{n-m_{j-1}}{s_j}(s_j!)\binom{n-m_{j-1}-s_j}{2c_j}\frac{(2c_j)!}{2^{c_j}}}{f_j^{n-m_{j-1}}} \times (n - m_{j-1} - s_j - 2c_j)^{c_j}. \tag{4.4}$$

Estimasi n adalah nilai yang memaksimalkan L. Biarkan nilai ini menjadi n, yang dapat ditemukan melalui pencarian lengkap karena rentang untuk n terbatas dalam praktik, jarang melampaui puluhan ribu. Untuk permintaan ke-i, kami mengatur ukuran bingkai menjadi n — $\mathrm{m_{j-1}}$.

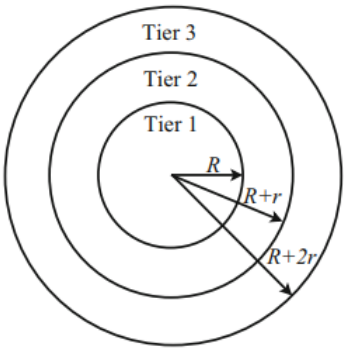
Penaksir di atas mengikuti prinsip umum yang awalnya terlihat pada, tetapi dibutuhkan informasi dari cj, sj, dan ej semua dalam penduga yang sama, sedangkan penduga pada menggunakan cj atau ej.

Seperti yang akan ditunjukkan oleh analisis kami, kecuali untuk pembaca, jumlah rata-rata anak per tag biasanya sangat kecil (kurang dari 2) untuk jaringan tag yang didistribusikan secara acak. Dalam hal ini, jika kita menyetel ukuran bingkai awal λ ke 4, kemungkinan besar tag berhasil mengumpulkan semua ID dari turunannya dalam kerangka waktu pertama. Oleh karena itu, hanya pembaca yang perlu menggunakan (4.4) untuk memperkirakan jumlah anak-anaknya, sedangkan tag hanya dapat mengatur ukuran bingkai ke konstanta kecil untuk menghindari overhead komputasi.

4.4.9 Faktor Beban Per Tag

Kami menganalisis beban kerja setiap tag dalam hal berapa banyak anak dan keturunan yang harus ditangani. Sementara pendekatan load balancing kami dirancang untuk setiap distribusi tag, untuk membuat analisis dapat dilakukan, kami berasumsi di sini bahwa tag didistribusikan secara merata di area dengan kepadatan p, dan tag yang jaraknya dari pembaca tidak lebih besar dari R membentuk tingkat pertama. , sedangkan yang jaraknya dari pembaca lebih besar dari R + (i – 2) tetapi lebih kecil dari R + (i – 1) membentuk tingkat jaringan ke-i (i $\geq$ 2), di mana rentang transmisi pembaca dan tag adalah R dan r, masing-masing, dengan R $\geq$ r. Sebagai contoh, Gambar 4.7 menyajikan jaringan dengan tiga tingkatan. Jumlah Ni tag di tingkat ke-i diperkirakan sebagai

$$\begin{aligned} N_i &= \rho \times (\pi \times (R + (i-1)r)^2 - \pi \times (R + (i-2)r)^2) \\ &= \pi\rho(2Rr + (2i-1)r^2). \end{aligned} \tag{4.5}$$



Gambar 4.7 Ilustrasi jaringan dengan tiga tingkatan tag

Tabel 4.2 Nilai Di dengan R = 3r

i	1	2	3	4	5	6	7	8	9
D_i	1.3	1.2	1.2	1.2	1.1	1.1	1.1	1.1	1.1

Satu pengecualian adalah bahwa N1 yang dihitung dari (4.5) sebenarnya hanya mencakup bagian dari tag tingkat-1 yang jaraknya dari pembaca lebih besar dari R – r; ini adalah tag yang dapat berfungsi sebagai induk untuk tag tier-2.

Derajat anak dari tag tier-i, dilambangkan dengan Di, didefinisikan sebagai jumlah rata-rata anak yang dimiliki tag tier-i. Karena tag di tingkat ke-i hanya berfungsi sebagai induk untuk tag di tingkat ke (i + 1), kita memiliki

$$D_i = \frac{N_{i+1}}{N_i} = \frac{2R + (2i+1)r}{2R + (2i-1)r} = 1 + \frac{1}{\frac{R}{r} + (i - \frac{1}{2})}. \tag{4.6}$$

Kami memiliki R ≫ r karena pembaca dapat mengirimkan pada tingkat daya yang jauh lebih tinggi dan memiliki antena yang jauh lebih sensitif. Hal ini membuat nilai D_i sangat kecil. Sebagai contoh, jika R 3r, Tabel 4.2 menunjukkan nilai Di, $1 \leq i < 10$, yang lebih kecil dari 1,3 dan cepat konvergen menuju 1 saat i meningkat. Nilai pada tabel akan semakin kecil jika R > 3r.

Faktor beban tag tier-i, dilambangkan sebagai L_i, didefinisikan sebagai jumlah rata-rata ID yang harus diteruskan oleh tag tier-i, termasuk ID turunannya tier- (i + 1) serta ID lain yang anak-anak mengumpulkan dari keturunan mereka. Li sama dengan jumlah total tag di luar tingkat ke-i dibagi dengan jumlah tag di tingkat ke-i.

$$\begin{aligned} L_i &= \frac{\sum_{j=i+1}^{l} N_j}{N_i} = \frac{\sum_{j=i+1}^{l} 2R + (2j-1)r}{2R + (2i-1)r} \\ &= \frac{2(l-i) + \frac{r}{R}(l^2 - i^2)}{2 + (2i-1)\frac{r}{R}}, \end{aligned} \tag{4.7}$$

di mana l adalah jumlah total tingkatan dan $i < l$. Ketika R = 3r dan l = 10, Tabel 4.3 menunjukkan nilai Li, $1 \leq i < 10$, yang sangat kecil. Karena tag tier-1 dapat ditenagai oleh gelombang radio

dari reader, kami hanya memperhatikan konsumsi daya tag di tier lain. Tag di tingkat 2 harus meneruskan lebih banyak ID daripada yang ada di tingkat luar. Dari tabel, tag tingkat-2 hanya meneruskan rata-rata 16 ID, yang merupakan overhead sederhana, mengingat ada delapan tingkatan lagi di luar tingkat 2. Meskipun rata-ratanya sederhana, faktor beban kasus terburuk juga penting saat kami mengevaluasi atas. SICP dirancang untuk mendistribusikan beban kerja secara merata di antara tag dengan menyeimbangkan pohon rentang, sehingga tag pada tingkat tertentu memiliki jumlah anak (atau keturunan) yang sama, yang diterjemahkan ke derajat anak (atau faktor beban) yang serupa. Kami akan mempelajari derajat anak-anak kasus terburuk dan faktor beban dengan simulasi.

Tabel 4.3 Nilai L_i dengan R = 3r dan l = 10

i	1	2	3	4	5	6	7	8	9
L_i	21.9	16.0	12.1	9.2	7.0	5.2	3.6	2.3	1.1

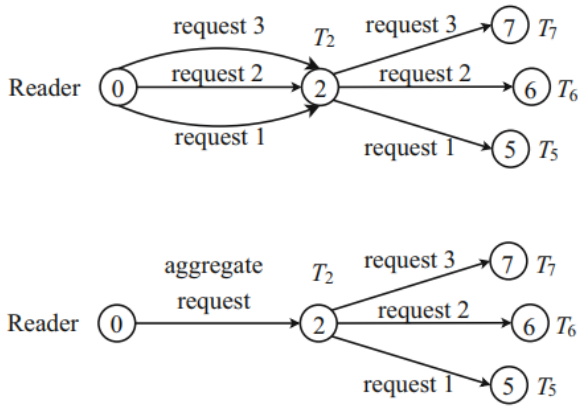
4.5 MENINGKATKAN EFISIENSI WAKTU SICP

Koleksi ID serial SICP menghilangkan sebagian besar transmisi simultan untuk tujuan mengurangi tabrakan, yang pada gilirannya dapat menurunkan efisiensi waktu. Di bagian ini, kami mengeksplorasi cara potensial untuk meningkatkan efisiensi waktu SICP. Proses tag untuk mengeksekusi SICP mencakup tiga langkah: (1) pembaca mengirimkan permintaan ke tag yang ditunjuk untuk melakukan pengumpulan ID; (2) tag yang ditunjuk mengumpulkan ID dari anak-anaknya; dan (3) ID yang dikumpulkan diteruskan ke pembaca. Pada langkah (2), ID dikumpulkan menggunakan protokol ALOHA slotted frame, dan ukuran frame optimal diberikan oleh (4.3). Dalam praktiknya, ukuran bingkai yang digunakan oleh tag dapat disetel ke konstanta kecil karena setiap tag hanya memiliki beberapa anak, sehingga mengurangi biaya komputasi tag untuk mengoptimalkan ukuran bingkai. Oleh karena itu, tujuan kami di sini adalah untuk meningkatkan efisiensi waktu langkah (1) dan langkah (3).

4.5.1 Permintaan Agregasi

Ingatlah bahwa setiap tag harus menerima permintaan RQST2 dari pembaca sebelum mengumpulkan ID dari turunannya. Permintaan diteruskan melalui beberapa lompatan di sepanjang jalur dari pembaca ke tag. Kami pertama-tama menggunakan contoh untuk mengilustrasikan ide agregasi permintaan. Perhatikan sebuah subpohon pada Gambar 4.3 yang terdiri dari pembaca, T_2, T_5, T_6, dan T_7. Seperti yang ditunjukkan di bagian atas Gambar. 4.8, permintaan ke T_5 harus diteruskan melalui dua pembaca hop $\rightarrow T_2$ dan $T_2 \rightarrow T_5$. Demikian pula, permintaan yang ditargetkan ke T_6 atau T_7 perlu diteruskan dua kali juga. Misalkan transmisi satu hop membutuhkan satu slot. Ini membutuhkan total enam slot untuk meneruskan ketiga permintaan ke T_5, T_6, dan T_7. Kami mengamati bahwa ketiga permintaan harus terlebih dahulu

diteruskan ke T_2, induk umum dari T_5, T_6, dan T_7. Pembaca! Transmisi T_2 dilakukan sebanyak tiga kali,



Gambar 4.8 Ilustrasi penggunaan agregasi permintaan untuk meningkatkan efisiensi waktu di SICP

Kami memang dapat menggabungkan tiga permintaan menjadi satu permintaan untuk menghindari transmisi yang berlebihan. Seperti yang ditunjukkan di bagian bawah Gambar 4.8, alih-alih mengirim perintah terpisah ke T_5, T_6, dan T_7 secara individual, pembaca pertama-tama mengirim permintaan agregat ke T_2, dan T_2 kemudian mengirim permintaan ke tiga anak untuk melakukan pengumpulan ID secara berurutan. Akibatnya, jumlah total slot untuk meneruskan permintaan dapat dikurangi menjadi empat. Dalam permintaan agregat, pembaca dapat memasukkan nomor seri anak-anak T_2 sehingga T_2 tidak perlu mengingat siapa anak-anaknya. Sebagai alternatif, kita dapat sedikit memodifikasi protokol SICP dengan meminta setiap tag untuk mencatat nomor seri awal dari turunannya dan jumlah turunan yang dimilikinya, yang dapat dengan mudah dicapai saat melakukan pengumpulan ID. Dengan dua nilai tersebut, setiap tag dapat memulihkan semua nomor seri turunannya bila diperlukan.

Misalkan T adalah tag tier-i dan memiliki m anak. Dalam desain asli SICP, dibutuhkan slot (i +1) untuk meneruskan permintaan RQST2 ke anak T, yang merupakan tag tier-(i+1). Oleh karena itu, total biaya waktu t_f untuk meneruskan permintaan ke setiap anak T adalah

$$t_f = (i + 1) \times m. \qquad (4.8)$$

Setelah menerapkan teknik agregasi permintaan, hanya satu permintaan agregat yang perlu dikirim ke T. Oleh karena itu, biaya waktu berkurang menjadi

$$t'_f = i + m. \qquad (4.9)$$

4.5.2 Pipa Transmisi ID

Setelah mengumpulkan semua ID dari turunannya, sebuah tag meneruskan ID yang dikumpulkan ke induknya, yang mungkin membutuhkan banyak slot. Hanya setelah menerima semua ID dari tag, Induk akan mulai meneruskan ID yang diterima tersebut. Proses ini berlanjut hingga ID akhirnya dikirimkan ke pembaca. Pertimbangkan tag tier-i. Misalkan memiliki m anak yang ID-nya dikumpulkan, dan k ID dapat ditransmisikan di setiap slot. Biaya waktu tb, dalam jumlah slot waktu yang diperlukan untuk meneruskan ID ke pembaca, kira-kira

$$t_b = i \times \frac{m}{k}. \qquad (4.10)$$

Cara pengiriman ID yang benar-benar serial ini, bagaimanapun, tidak efisien waktu karena hanya satu tag yang diizinkan untuk dikirimkan setiap saat. Kami ingin mengeksploitasi transmisi simultan di antara tag yang tidak mengganggu melalui penggunaan kembali saluran spasial. Sebelum memperkenalkan ide transmisi pipelining, terlebih dahulu kita buktikan teorema berikut:

Teorema 8. Jika tag T adalah ancestor node tetapi bukan parent node langsung dari tag T^0 dalam spanning tree yang dibangun oleh SICP, T^0 tidak boleh menjadi tetangga dari T.

Bukti. Dibuktikan dengan kontradiksi, kita asumsikan bahwa T^0 adalah tetangga dari T. Tunjukkan induk dari T^0 sebagai Tp. Ingatlah bahwa sebuah tag akan menentukan induknya saat menerima permintaan pengumpulan ID pertama. Misalkan t0 adalah waktu ketika T^0 menentukan induknya, dan t adalah waktu ketika T menyiarkan permintaan pengumpulan ID untuk pertama kalinya. Karena T^0 mungkin mendengar permintaan untuk pertama kalinya dari node lain, kita harus memiliki

$$t' \leq t. \qquad (4.11)$$

Biarkan tp menjadi waktu ketika Tp berhasil mengirimkan ID-nya. Karena T juga merupakan node ancestor dari Tp, pengiriman ID Tp harus terjadi setelah T mengirimkan permintaan koneksi ID-nya. Oleh karena itu, t < tp. Dari (4.11), kita memiliki

$$t' < t_p. \qquad (4.12)$$

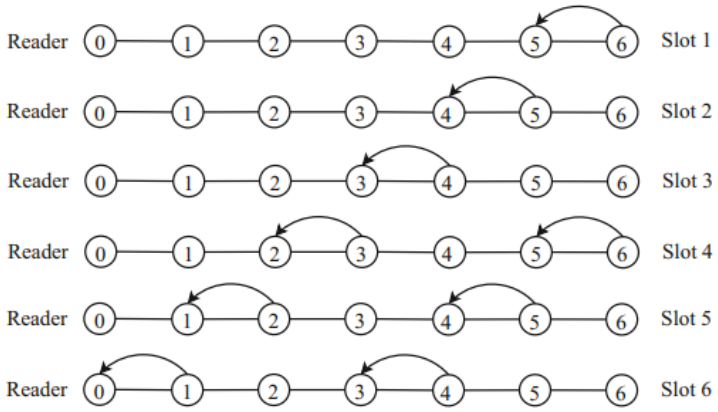
Kita tahu bahwa Tp dipilih sebagai induk dari T^0 pada waktu t0 (setelah menerima permintaan pengumpulan ID). Untuk memilih Tp sebagai induknya, T harus mengetahui keberadaan Tp lebih awal, yang terjadi pada saat tp (saat Tp berhasil memberikan ID-nya ke induknya). Yaitu, perlu bahwa yang bertentangan dengan (4.12) dan dengan demikian melengkapi bukti.

$$t_p < t', \qquad (4.13)$$

Berdasarkan Teorema 8, kita dapat menyimpulkan bahwa untuk transmisi ID arbitrer dari T00 ke T, transmisi dari salah satu node ancestor T yang bukan induknya tidak akan menyebabkan gangguan pada penerimaan transmisi T oleh T^{00}. Kami menggunakan contoh pada Gambar 4.9

untuk mengilustrasikan ide pipelining transmisi ID. Misalkan node 6 telah mengumpulkan ID dari anak-anaknya dan mulai meneruskan ID ke pembaca di sepanjang jalur 6 → 5 → 4 → 3 → 2 → 1 → 0. Alih-alih meneruskan semua ID yang dikumpulkan ke node induknya 5 sekaligus, yang merupakan metode yang diadopsi oleh SICP, node 6 pertama-tama hanya menggunakan satu slot untuk mengirimkan beberapa ID (dengan asumsi mereka tidak dapat masuk semua dalam satu slot) ke induknya, seperti yang ditunjukkan oleh slot 1 pada Gambar 4.9.

Pada dua slot berikut, hanya node 5 dan node 4 yang meneruskan ID yang diterima, sedangkan node 6 tidak melakukan transmisi apapun untuk menghindari tabrakan. Pada slot 4, ketika node 3 meneruskan ID yang diterima ke node 2, node 6 akan mengirimkan slot ID lainnya ke node 5. Menurut Teorema 8, node 3 bukan tetangga dari node 5, dan dengan demikian transmisinya tidak akan mengganggu node 5 menerima transmisi dari node 6. Singkatnya, node 6 dapat melakukan satu transmisi setiap tiga slot dalam contoh ini. Hal ini juga berlaku untuk node lain di jalur dari node 6 ke reader. Transmisi paralel secara efektif menghasilkan pipa transmisi dalam mengirimkan ID ke pembaca.



Gambar 4.9 Transmisi pipelining dari ID yang dikumpulkan oleh node 6. Setiap panah mewakili satu transmisi ID dari node ke induknya di slot yang diberikan

Kami menggeneralisasi teknik pemipaan transmisi untuk setiap tag dalam sistem sebagai berikut:

1. Setelah tag tier-1 mengumpulkan ID dari anak-anaknya, tag akan langsung mengirimkan ID ke pembaca dalam slot terus menerus tanpa menunggu.
2. Setelah tag tier-2 T mengumpulkan ID dari anak-anaknya, tag melakukan satu transmisi ID setiap dua slot, memungkinkan induknya untuk meneruskan ID yang diterima ke pembaca di slot waktu segera setelah setiap slot ketika T mentransmisikan.
3. Setelah tag T di tier 3 atau lebih tinggi mengumpulkan ID dari turunannya, tag melakukan satu transmisi ID setiap tiga slot untuk mendukung pipelining. Ketika setiap tag di jalur dari T ke pembaca menerima ID di slot, itu akan mengirimkan ID yang diterima di slot berikutnya.

Oleh karena itu, biaya waktu t_b^0 untuk tag tier-i untuk meneruskan m ID yang dikumpulkan ke pembaca dengan pipa transmisi kira-kira

$$t_b' = \begin{cases} i \times \frac{m}{k} & \text{if } 1 \le i \le 2 \\ 3 \times (\frac{m}{k} - 1) + i & \text{if } i \ge 3 \end{cases} \qquad (4.14)$$

Sangat mudah untuk membuktikan bahwa $t_b^0 \le t_b$. Pipa transmisi dapat meningkatkan waktu efisiensi pengumpulan ID, terutama untuk tag dengan nomor tier besar. Kami menyatakan SICP dengan agregasi permintaan dan pipa transmisi sebagai p-SICP dalam sekuel.

4.6 EVALUASI

4.6.1 Pengaturan Simulasi

Tidak ada pekerjaan sebelumnya pada identifikasi tag untuk sistem tag jaringan. Tetapi teknik yang dikenal seperti transmisi berbasis siaran dan konten yang banyak digunakan dalam sistem nirkabel lain dapat digunakan untuk merancang protokol identifikasi tag bebas-negara, CICP, yang akan kita gunakan sebagai bahan pembanding. Kami mengevaluasi kinerja CICP, SICP, dan p-SICP untuk menunjukkan tiga temuan utama bahwa (1) meskipun protokol berbasis ALOHA sangat berhasil di sistem nirkabel lain (termasuk sistem RFID), mereka tidak cocok untuk sistem tag jaringan, dan bahwa (2) serialisasi dapat secara signifikan meningkatkan kinerja identifikasi tag, dan (3) teknik agregasi permintaan dan pipelining transmisi ID dapat secara signifikan meningkatkan efisiensi waktu pengumpulan ID serial.

Tiga metrik kinerja digunakan: (1) waktu eksekusi yang diukur dalam jumlah slot waktu, (2) jumlah bit rata-rata dan maksimum yang dikirim per tag, dan (3) jumlah bit rata-rata dan maksimum yang diterima per tag. Dua yang terakhir adalah ukuran tidak langsung dari biaya energi, di mana tag tier-1 dikecualikan karena dapat ditenagai oleh gelombang radio pembaca. Perhitungan dengan tag dalam dua protokol sangat terbatas. Sebagian besar energi dihabiskan untuk komunikasi. Jumlah data komunikasi berfungsi sebagai sarana tidak langsung untuk membandingkan protokol yang berbeda. Misalnya, jika tag dalam satu protokol menerima dan mengirim jauh lebih banyak daripada yang ada di protokol lain, aman untuk mengatakan bahwa protokol pertama membutuhkan lebih banyak energi daripada yang kedua.

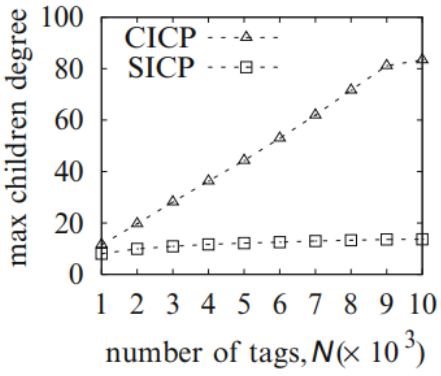
Kami memvariasikan jumlah N tag dalam sistem dari 1000 hingga 10.000 pada langkah 1000. Tag didistribusikan secara acak di area melingkar dengan radius 50 m kecuali jika parameter eksplisit ditentukan. Pembaca, yang jangkauan komunikasinya R diatur ke 25 m, terletak di tengah area. Untuk setiap tag, jarak komunikasi antar tag r adalah 5 m. Dalam SICP dan p-SICP, reader menyetel ukuran frame dari permintaan ke-i ke f_i = maks{n m_{i-1}, f_l}, di mana m_{i-1} adalah jumlah ID yang telah dikumpulkan dan n adalah jumlah perkiraan tag yang memaksimalkan (4.4). Batas bawah f_l, ditetapkan ke 50, mencegah ukuran bingkai diatur terlalu kecil atau bahkan negatif karena penyimpangan estimasi n.

Ukuran bingkai awal λ adalah 50 untuk pembaca. Untuk membebaskan tag dari memperkirakan jumlah anak yang mereka miliki, kami membiarkan mereka menggunakan ukuran bingkai tetap λ dengan nilai default 4, tetapi kami juga akan memvariasikannya dari 2 hingga 10. Panjang setiap ID tag adalah 96 bit. . Panjang setiap nomor seri adalah $[\log_2 N]$ bit. Panjang setiap nomor tier adalah 4 bit. Mengikuti spesifikasi standar EPC global Class-1 Gen-2, kami menetapkan panjang semua jenis permintaan menjadi 20 bit, dan mengatur ACK dan NAK masing-masing menjadi 16 bit dan 8 bit. Dalam SICP dan p-SICP, ACK juga akan menyertakan nomor seri. Untuk setiap titik data dalam gambar, kami mengulangi simulasi sebanyak 100 kali dan menyajikan hasil rata-rata.

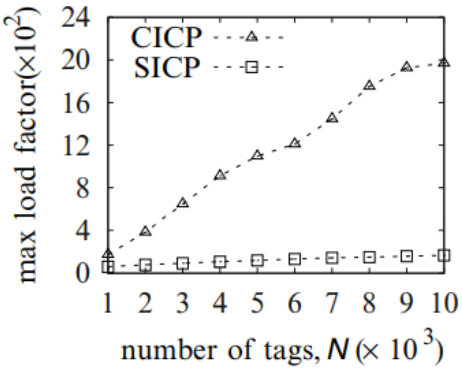
4.6.2 Derajat Anak dan Faktor Beban

Kami pertama memeriksa keseimbangan pohon rentang yang dibangun oleh CICP dan SICP (pohon rentang di p-SICP dibangun dengan cara yang sama seperti SICP). Ini memiliki dampak yang signifikan pada biaya energi kasus terburuk dari tag. Tag dengan derajat turunan yang lebih besar (atau faktor beban yang lebih besar) harus mengumpulkan (atau meneruskan) lebih banyak ID tag, yang menghasilkan pengeluaran energi tambahan. Tag yang memiliki derajat turunan atau faktor beban terbesar dapat menjadi penghambat energi dalam jaringan. Jika energi sisa pada tag habis sebelum protokol selesai, jaringan bahkan dapat dipartisi karena tag mati.

Gambar 4.10 dan 4.11 menyajikan derajat anak maksimum dan faktor beban maksimum di pohon merentang yang dibangun oleh CICP dan SICP, masing-masing. Karena jumlah N tag dalam sistem menjadi lebih besar, peningkatan angka kasus terburuk ini di bawah CICP jauh lebih cepat daripada peningkatan di bawah SICP, menunjukkan pohon yang jauh lebih seimbang untuk yang terakhir. Misalnya, ketika N 10.000, derajat anak maksimum dan faktor beban di CICP adalah 83 dan 1969, dan angka-angka di SICP hanya 14 dan 165.



Gambar 4.10 Derajat anak maksimum dalam pohon merentang

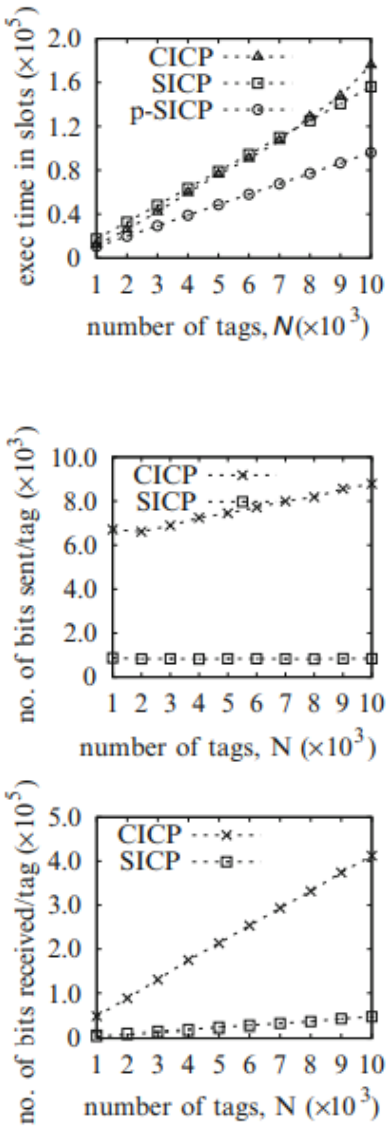


Gambar 4.11 Faktor beban maksimum tag di pohon rentang

4.6.3 Perbandingan Kinerja

Kami membandingkan kinerja CICP, SICP, dan p-SICP pada Gambar 4.12, di mana plot pertama menunjukkan waktu eksekusi protokol dalam hal jumlah slot yang digunakan, plot kedua menunjukkan jumlah rata-rata bit yang dikirim per tag, dan plot ketiga menunjukkan jumlah rata-rata bit yang diterima per tag. SICP menggunakan jumlah slot yang sebanding sebagai CICP, sedangkan p-SICP membutuhkan jumlah slot yang jauh lebih kecil daripada SICP, seperti yang kami harapkan. Biaya energi SICP dan p-SICP sangat dekat. Keduanya jauh lebih kecil daripada CICP, berkat serialisasi untuk pengurangan tabrakan. Misalnya, ketika N = 10.000, jumlah bit yang dikirim/diterima per tag di CICP adalah 8783 dan 412.218, sedangkan angka-angka tersebut masing-masing hanya 862 dan 54.871 untuk SICP, yang mewakili pengurangan 90,2 dan 86,7% dari CICP. Karena teknik agregasi permintaan, jumlah rata-rata bit yang dikirim per tag di p-SICP sedikit lebih kecil daripada SICP. Tetapi setiap tag di p-SICP menerima bit sedikit lebih banyak daripada rata-rata SICP. Alasannya adalah bahwa tag di SICP dapat memberi tahu tetangga non-orangtuanya untuk tidur selama durasi tertentu tanpa menerima ID yang tidak perlu; sebaliknya, perpipaan transmisi p-SICP mengharuskan setiap tetangga menerima apa yang ditransmisikan oleh tag di setiap slot. Untuk p-SICP, jumlah bit yang dikirim/diterima per tag adalah 634 dan 63.716, yang masing-masing mewakili pengurangan 92,8 dan 84,5% dari CICP.

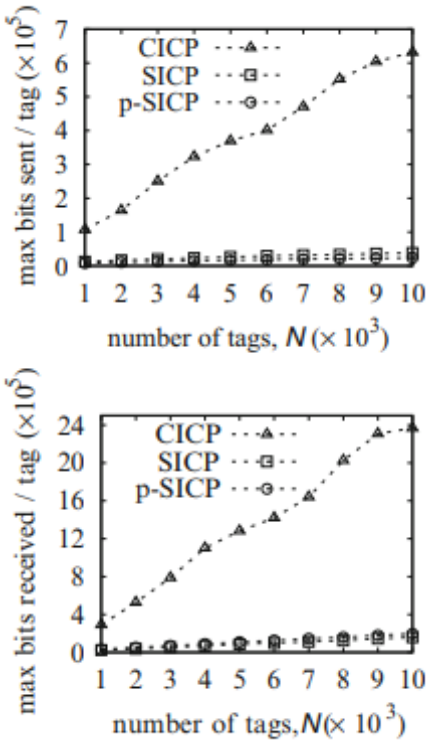
Gambar 4.13 menunjukkan jumlah maksimum bit yang dikirim/diterima oleh tag di bawah tiga protokol, masing-masing. Seperti yang diharapkan, tag yang paling memakan energi menghabiskan lebih sedikit energi di bawah SICP dan p-SICP daripada di bawah CICP. Misalnya, ketika N 10,000, jumlah bit maksimum yang dikirim/diterima oleh tag apa pun di CICP adalah 631.412 dan 2.367.899, dan angka-angka di SICP adalah 38.273 dan 159.431—93,9% dan 93,3% pengurangan, masing-masing.



Gambar 4.12 Perbandingan kinerja antara CICP dan SICP

4.6.4 Pengorbanan Kinerja untuk SICP dan p-SICP

Selanjutnya, kami mendemonstrasikan *tradeoff* kinerja untuk SICP dan p-SICP yang dikendalikan oleh nilai h. Kami menetapkan N = 5000 dan memvariasikan h dari 2 hingga 10. Hasilnya disajikan pada Gambar 4.14, di mana tiga plot dari kiri ke kanan menunjukkan waktu eksekusi, jumlah rata-rata bit yang dikirim per tag, dan jumlah rata-rata bit yang diterima per tag, masing-masing. *Tradeoff* waktu-energi yang serupa dapat diamati pada SICP dan p-SCIP. Ketika nilai h meningkat, waktu eksekusi meningkat, tetapi biaya energi untuk mengirim dan menerima berkurang. Namun, waktu meningkat hampir secara linier, tetapi penurunan energi mendatar, menunjukkan bahwa nilai h yang sederhana lebih disukai.



Gambar 4.13 Plot atas: jumlah bit maksimum yang dikirim oleh tag apa pun dalam sistem. Plot bawah: jumlah bit maksimum yang diterima oleh tag apa pun di sistem

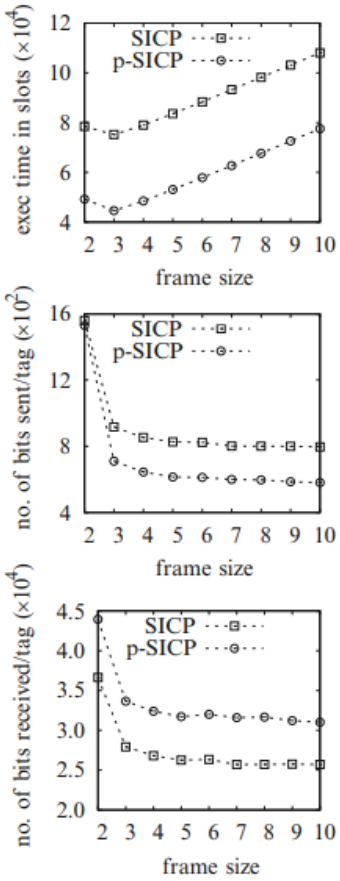
4.6.5 Perbandingan Efisiensi Waktu SCIP dan p-SICP

Analisis di bab 4.5 menunjukkan bahwa teknik agregasi permintaan dan pipelining transmisi dapat mengurangi waktu pengumpulan ID, terutama untuk tag dengan nomor tier yang besar. Kami menggunakan simulasi untuk memverifikasi kesimpulan ini. Kami memvariasikan radius area melingkar, di mana 5.000 tag jaringan didistribusikan secara acak, dari 50 hingga 100 m pada langkah 10 m. Radius yang lebih besar dari area distribusi berarti ada lebih banyak tingkatan dalam sistem, menghasilkan ketinggian yang lebih besar dari pohon rentang. Gambar 4.15 membandingkan waktu eksekusi SICP dan p-SICP. Kita dapat melihat bahwa gap antara waktu eksekusi SICP dan p-SICP menjadi lebih besar dengan bertambahnya radius. Ketika radiusnya besar, p-SICP memotong waktu eksekusi SICP lebih dari setengahnya.

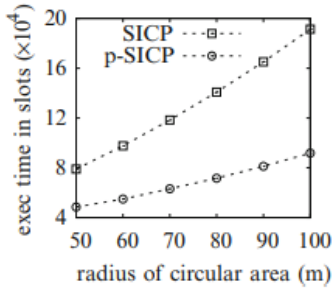
4.7 RINGKASAN

Bab ini membahas masalah identifikasi tag dalam sistem tag jaringan yang muncul. Sifat multihop dari sistem tag jaringan membuat masalah ini berbeda dari masalah identifikasi tag dalam sistem RFID. Dua protokol identifikasi tag dirancang dengan tiga temuan penting. Temuan pertama adalah bahwa desain protokol berbasis pertentangan tradisional menimbulkan terlalu banyak energi overhead dalam sistem tag jaringan karena tabrakan yang

berlebihan. Temuan kedua adalah bahwa ketidakseimbangan beban menyebabkan biaya energi terburuk yang besar pada tag. Kami mengatasi masalah ini melalui serialisasi dan pemilihan induk probabilistik berdasarkan nomor seri. Temuan ketiga adalah bahwa teknik agregasi permintaan dan pipelining transmisi ID dapat secara signifikan meningkatkan efisiensi waktu pengumpulan ID serial.



Gambar 4.14 Waktu eksekusi dan biaya energi SICP dan p-SICP terhadap λ, ketika N = 5000



Gambar 4.15 Perbandingan waktu eksekusi SICP dan

p- SICP ketika 5000 tag jaringan didistribusikan secara acak di atas area melingkar dengan radius berbeda.

DAFTAR PUSTAKA

AEI Technology. Available at http://www.aeitag.com/aeirfidtec.html (2008)

Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash functions and RFID tags: mind the gap. In: Proceedings of CHES, pp. 283-299 (2008)

Broder, A., Mitzenmacher, M.: Network applications of bloom filters: a survey. Internet Math. 1(4), 485-509 (2003)

Bu, K., Xiao, B., Xiao, Q., Chen, S.: Efficient pinpointing of misplaced tags in large RFID systems. In: Proceedings of IEEE SECON, pp. 287-295 (2011)

Castiglione, P., Ricciato, F., Popovski, P.: Pseudo-random Aloha for inter-frame soft combining in RFID systems. In: Proceedings of IEEE DSP, pp. 1-6 (2013)

Cha, J.R., Kim, J.H.: Dynamic framed slotted ALOHA algorithms using fast tag estimation method for RFID systems. In: Proceedings of IEEE Consumer Communications and Networking Conference (CCNC) (2006)

Chen, M., Chen, S., Xiao, Q.: Pandaka: a lightweight cipher for RFID systems. In: Proceedings of IEEE INFOCOM, pp. 172-180 (2014)

Chen, M., Chen, S.: An efficient anonymous authentication protocol for RFID systems using dynamic tokens. In: Proceedings of IEEE ICDCS (2015)

Chen, M., Chen, S.: ETAP: enable lightweight anonymous RFID authentication with O(1) overhead. In: Proceedings of IEEE ICNP (2015)

Chen, M., Chen, S.: Identifying state-free networked tags. In: Proceedings of IEEE ICNP (2015)

Chen, M., Luo, W., Mo, Z., Chen, S., Fang, Y.: An efficient tag search protocol in large-scale RFID systems with noisy channel. IEEE/ACM Trans. Networking PP(99), 1-1 (2015)

Chen, M., Luo, W., Mo, Z., Chen, S., Fang, Y.: An efficient tag search protocol in large-scale RFID systems. In: Proceedings of IEEE INFOCOM, pp. 1325-1333 (2013)

Chen, S., Deng, Y., Attie, P., Sun, W.: Optimal deadlock detection in distributed systems based on locally constructed wait-for graphs. In: Proceedings of IEEE INFOCOM, pp. 613-619 (1996)

Chen, S., Zhang, M., Xiao, B.: Efficient information collection protocols for sensor-augmented RFID networks. In: Proceedings of IEEE INFOCOM, pp. 3101-3109 (2011)

Choi, J., Lee, C.: A cross-layer optimization for a LP-based multi-reader coordination in RFID systems. In: Proceedings of IEEE GLOBECOM, pp. 1-5 (2010)

Cornaglia, B., Spini, M.: New statistical model for burst error distribution. Eur. Trans. Telecommun. 7, 267-272 (1996)

Dan, L., Wei, P., Wang, J., Tan, J.: TFDMA: a scheme to the RFID reader collision problem based on graph coloration. In: Proceedings of IEEE SOLI, pp. 502-507 (2008)

Eom, J., Lee, T.: Accurate tag estimation for dynamic framed-slotted ALOHA in RFID systems. In: Proceedings of IEEE Communication Letters, pp. 60-62 (2010)

EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communi- cations at 860MHz-960MHz, EPCglobal. Available at http://www.epcglobalinc.org/uhfclg2 (2011)

EPC Radio-Frequency Identity Protocols Class-1 Gen-2 UHF RFID Protocol for Communi- cations at 860MHz-960MHz, EPCglobal (2011). Available at http://www.epcglobalinc.org/ uhfclg2

Federal Standard 1037C. Available at http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm (1996)

Fletcher, R., Marti, U.P., Redemske, R.: Study of UHF RFID signal propagation through complex media. In: IEEE Antennas and Propagation Society International Symposium, vol. 1B, pp. 747-750 (2005)

Fyhn, K., Jacobsen, R.M., Popovski, P., Scaglione, A., Larsen, T.: Multipacket reception of passive UHF RFID tags: a communication theoretic approach. IEEE Trans. Signal Process. 59(9), 4225-4237 (2011)

Gorlatova, M., Kinget, P., Kymissis, I., Rubenstein, D., Wang, X., Zussman, G.: Challenge: ultra-low-power energy-harvesting active networked tags (EnHANTs). In: Proceedings of ACM Mobicom, pp. 253-260 (2009)

Gorlatova, M., Margolies, R., Sarik, J., Stanje, G., Zhu, J., Vigraham, B., Szczodrak, M., Carloni, L., Kinget, P., Kymissis, I., Zussman, G.: Prototyping energy harvesting active networked tags (EnHANTs). In: Proceedings of IEEE INFOCOM Mini-Conference (2013)

Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight Hash functions. In: Proceedings of CRYPTO, pp. 222-239 (2011)

Han, H., Sheng, B., Tan, C.C., Li, Q., Mao, W., Lu, S.: Counting RFID tags efficiently and anonymously. In: Proceedings of IEEE INFOCOM (2010)

Johnson, N.L., Kotz, S.: Urn Models and Their Application: An Approach to Modern Discrete Probability Theory. Wiley, New York (1977)

Kaitovic, J., Langwieser, R., Rupp, M.: A smart collision recovery receiver for RFIDs. EURASIP J. Embed. Syst. 2013, 1-19 (2013)

Kaitovic, J., Rupp, M.: Improved physical layer collision recovery receivers for RFID readers. In: Proceedings of IEEE RFID, pp. 103-109 (2014)

Kang, Y., Kim, M., Lee, H.: A hierarchical structure based reader anti-collision protocol for dense RFID reader networks. In: Proceedings of ICACT, pp. 164-167 (2011)

Kinget, P., Kymissis, I., Rubenstein, D., Wang, X., Zussman, G.: Energy harvesting active networked tags (EnHANTs) for ubiquitous object networking. IEEE Trans. Wirel. Commun. 17(6), 18-25 (2010)

Kodialam, M., Nandagopal, T.: Fast and reliable estimation schemes in RFID systems. In: Proceedings of ACM MobiCom (2006)

Kong, L., He, L., Gu, Y., Wu, M., He, T.: A parallel identification protocol for RFID systems. In: Proceedings of IEEE INFOCOM, pp. 154-162 (2014)

Kronecker delta. Available at http://en.wikipedia.org/wiki/Kronecker_delta

Lee, C.H., Chung, C.W.: Efficient storage scheme and query processing for supply chain management using RFID. In: Proceedings of ACM SIGMOD (2008)

Lee, S., Joo, S., Lee, C.: An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification. In: Proceedings of IEEE MobiQuitous (2005)

Li, T., Chen, S., Ling, Y.: Efficient protocols for identifying the missing tags in a large RFID system. IEEE/ACM Trans. Networking 21(6), 1974-1987 (2013)

Li, Y., Ding, X.: Protecting RFID communications in supply chains. In: Proceedings of IEEE ASIACCS (2007)

Liu, J., Chen, M., Xiao, B., Zhu, F., Chen, S., Chen, L.: Efficient RFID grouping protocols. IEEE/ACM Trans. Networking PP(99), 1-1 (2016)

Liu, J., Xiao, B., Bu, K., Chen, L.: Efficient distributed query processing in large RFID-enabled supply chains. In: Proceedings of IEEE INFOCOM, pp. 163-171 (2013)

Liu, V., Parks, A., Talla, V., Gollakota, S., Wetherall, D., Smith, J.R.: Ambient backscatter: wireless communication out of thin air. In: Proceedings of ACM SIGCOMM, pp. 39-50 (2013)

Luo, W., Chen, S., Li, T.: Probabilistic missing-tag detection and energy-time tradeoff in large-scale RFID systems. In: Proceedings of ACM Mobihoc (2012)

Luo, W., Qiao, Y., Chen, S., Chen, M.: An efficient protocol for RFID multigroup threshold- based classification based on sampling and logical bitmap. IEEE/ACM Trans. Networking 24(1), 397-407 (2016)

Luo, W., Qiao, Y., Chen, S.: An efficient protocol for RFID multigroup threshold-based classification. In: Proceedings of IEEE INFOCOM, pp. 890-898 (2013)

Myung, J., Lee, W.: Adaptive splitting protocols for RFID tag collision arbitration. In: Proceedings of ACM MOBIHOC (2006)

Network Everything. Available at http://openinterconnect.org

Nguyen, C.T., Hayashi, K., Kaneko, M., Popovski, P., Sakai, H.: Probabilistic dynamic framed slotted ALOHA for RFID tag identification. Wirel. Pers. Commun. 71, 2947-2963 (2013)

Ni, L., Liu, Y., Lau, Y.C.: Landmarc: indoor location sensing using active RFID. In: Proceed- ings of IEEE PerCom (2003)

NIST: RFID Communication and Interference. White paper, Grand Prix Application Series (2007)

Onat, I., Miri, A.: A tag count estimation algorithm for dynamic framed ALOHA based RFID MAC protocols. In: Proceedings of IEEE ICC, pp. 1-5 (2011)

O'Neill, M.: Low-cost SHA-1 hash function architecture for RFID tags. In: Proceedings of RFIDSec (2008)

Qian, C., Liu, Y., Ngan, H., Ni, L.M.: ASAP: scalable identification and counting for contactless RFID systems. In: Proceedings of IEEE ICDCS (2010)

Qiao, Y., Chen, S., Li, T.: Energy-efficient polling protocols in RFID systems. In: Proceedings of ACM Mobihoc (2011)

Qiao, Y., Li, T., Chen, S.: One memory access bloom filters and their generalization. In: Proceedings of IEEE INFOCOM, pp. 1745-1753 (2011)

RFID Report. Available at http://www.idtechex.com/research/reports/rfid-forecasts-players-and-opportunities-2014-2024-000368.asp (2013)

Ricciato, F., Castiglione, P.: Pseudo-random ALOHA for enhanced collision-recovery in RFID. IEEE Commun. Lett. 17(3), 608-611 (2013)

Schoute, F.C.: Dynamic frame length ALOHA. IEEE Trans. Commun. 31, 565-568 (1983)

Shahzad, M., Liu, A.: Every bit counts - fast and scalable RFID estimation. In: Proceedings of ACM Mobicom (2012)

Shahzad, M., Liu, A.X.: Probabilistic optimal tree hopping for RFID identification. In: Proceedings of ACM SIGMETRICS, pp. 293-304 (2013)

Sheng, B., Li, Q., Mao, W.: Efficient continuous scanning in sistem RFID. Dalam: Prosiding IEEE INFOCOM (2010)

Sheng, B., Tan, C., Li, Q., Mao, W.: Finding popular categories for RFID tags. In: Proceedings of ACM Mobihoc (2008)

SINIAV. Tersedia di http://roadpricing.blogspot.com/2012/08/brazil-to-have-compulsory- toll-tags-by.html (2012)

Stefanovic, C., Popovski, P.: ALOHA random access that operates as a rateless code. IEEE Trans. Commun. 61(11), 4653-4662 (2013)

Sun Pass. Tersedia di https://www.sunpass.com/index

Xia, Y., Chen, S., Cho, C., Korgaonkar, V.: Algoritma dan kinerja load-balancing dengan beberapa fungsi hash dalam distribusi konten besar-besaran. Hitung. jaringan 53(1), 110-125 (2009)

Xiao, Q., Chen, M., Chen, S., Zhou, Y.: Estimasi RFID bersama yang tersebar secara temporal atau spasial menggunakan snapshot dari panjang variabel. Dalam: Prosiding ACM Mobihoc (2015)

Xiao, Q., Chen, S., Chen, M.: Estimasi properti bersama untuk beberapa set tag RFID menggunakan snapshot dari panjang variabel. Dalam: Prosiding ACM Mobihoc (2016)

Zhang, M., Li, T., Chen, S., Li, B.: Using analog network coding to improve the RFID reading throughput. In: Proceedings of IEEE ICDCS (2010)

Zhang, Z., Chen, S., Ling, Y., Chow, R.: Algoritma multicast yang sadar kapasitas pada jaringan overlay heterogen. IEEE Trans. Distribusi Paralel. sistem 17(2), 135-147 (2006)

Zheng, Y., Li, M.: Fast tag searching protocol for large-scale RFID systems. IEEE/ACM Trans. Networking 21(3), 924-934 (2012)

Zheng, Y., Li, M.: Protokol pencarian tag cepat untuk sistem RFID skala besar. IEEE/ACM Trans. Jaringan 21(3), 924-934 (2012)