Traffic analysis resistance

Tom Ristenpart
CS 6431
Onion routing (low-latency)
Onion routing (low-latency)

At least: traffic correlation attacks
  • Correlate timing of packets sent in from 1.2.3.4 and those received at 5.6.7.8
Onion routing (low-latency)

Many suggestions:
• adding noise (dummy requests/traffic) to obfuscate traffic patterns. Ad hoc suggests subsequently (academically) broken
Each message onion-encrypted
- All messages must be padded to the same length
- First mix node waits for lots of encrypted messages
  - Decrypts outer layer, shuffles, sends to the next node
- Final node can send messages to destinations
- Security should hold if any single node trustworthy

\[ E_{pk_1}(E_{pk_2}(E_{pk_3}(\text{Bob} \ || \ M))) \]
All-but-one traffic analysis threat model

- Adversary controls all but one server
- Adversary can monitor, block, delay, inject traffic on any network link
  - Adversary knows all users that participate
What is protected? What is leaked?
• Set of users who sent a message
• Set of users who received a message

\[ E_{pk1}(E_{pk2}(E_{pk3}(\text{Bob} \ || \ E_{pkBob}(M)))) \]

Have all users always send a message (can be dummy)

Don’t reveal recipient in final plaintext. All users download all final ciphertexts. Trial decrypt
Mixnets

$E_{pk1}(E_{pk2}(E_{pk3}(E_{pkBob}(M))))$

What is protected? What is leaked?
- Set of users who sent a message
- Set of users who received a message

Have all users always send a message (can be dummy)

Don’t reveal recipient in final plaintext. All users download all final ciphertexts. Trial decrypt
Private information retrieval

Can Bob recover his ciphertext without downloading every ciphertext?
Chor, Goldreich, Kushilevitz and Sudan introduced PIR in 1995

\[(1, C_1)\]
\[(2, C_2)\]
\[\ldots\]
\[(u, C_u)\]

Server learns nothing about \(i\)

**Requires** \(O(u)\) work on server

Bob learns \(C_i\)

Use less than \(O(u)\) bandwidth?

Two variants:
- Information-theoretic (IT-PIR): Split database across \(k\) servers. As long as one is honest, adversary can’t learn anything about \(i\)
- Computational (CPIR): Single computationally-bounded database can’t learn anything about \(i\)
Basic IT-PIR scheme

Can Bob recover his ciphertext without downloading every ciphertext?

\[(1, C_1) \quad (2, C_2) \quad \ldots \quad (u, C_u)\]

\[Y = R \oplus X_i\]

\[Z_1 = \bigoplus_{Y[j]=1} C_j\]

\[Z_2 = \bigoplus_{R[j]=1} C_j\]

If servers don’t collude, either learns nothing about \(i\)

\[C_i = Z_1 \oplus Z_2\]

\[R \leftarrow \{0,1\}^u\]

\[X_i = [0,...,0,1,0,...]\]

\(i^{th}\) bit is set to 1
Basic CPIR scheme

Uses homomorphic encryption: \( Enc(k,m_1) \times Enc(k,m_2) = Enc(k,m_1 + m_2) \)

\[
Z = \prod_{j=1}^{u} Q_j^{C_j}
\]

\( Q_j = Enc(k,0) \quad j \neq i \)
\( Q_j = Enc(k,1) \quad j = i \)

\( C_j = Dec(k,Z) \)

\( j \neq i : \quad Q_j^{C_j} = Enc(k,0) \quad C_j \text{ times} \)

\( j = i : \quad Q_j^{C_j} = Enc(k,1) \times Enc(k,1) \times \cdots \times Enc(k,1) \)
\[= Enc(k, C_j) \]
Basic CPIR scheme

Uses homomorphic encryption: \( \text{Enc}(k,m_1) \times \text{Enc}(k,m_2) = \text{Enc}(k,m_1 + m_2) \)

\[
(1,C_1) \\
(2,C_2) \\
... \\
(u,C_u)
\]

\[ Q_1, ..., Q_u \]

\[ i \]

\[ Q_j = \text{Enc}(k,0) \quad j \neq i \]
\[ Q_j = \text{Enc}(k,1) \quad j = i \]

Security: as long as \( \text{Enc} \) is IND-CPA, no computationally bound adversary can determine \( i \).
Fast CPIR Schemes

- XPIR scheme based on ring LWE (lattices)
  - Aguilar-Melchor et al. 2014

100 Gb database processed in a few seconds
Mixnets

What is protected? What is leaked?
• Set of users who sent a message
• Set of users who received a message

Have all users always send a message (can be dummy)

Don’t reveal recipient in final plaintext. All users download all final ciphertexts. Trial decrypt
Mixnets

\[ E_{pk1}(E_{pk2}(E_{pk3}(E_{pkBob}(M)))) \]

Recipients can each use CPIR to retrieve their ciphertext?

How does Bob know what index i his ciphertext is in?
Mixnets

Recipients can each use CPIR to retrieve their ciphertext?
How does Bob know what index his ciphertext is in?
Mixnets

$E_{pk1}(E_{pk2}(E_{pk3}(F_K(Bob) || E_{pkBob}(M))))$

Replace indices $i$ with $T_i = F_K(Bob)$ for PRF $F$ and key $K$ known to Alice and Bob.
Do PIR over compact data structure representation of this table

- Pung [Angel, Setty 2016] uses binary search trees
No-trust* traffic analysis threat model

- Adversary controls \textit{all} servers
- Adversary can monitor, block, delay, inject traffic on any network link
  - Adversary knows all users that participate

* Still need to trust developers, other communication partners, end-point security, etc.
PIR-based schemes

Get rid of mixnets entirely
Security holds even if server is adversarial
• Still need every client to always send messages
• Need to never reuse tags (add counters)
“Perhaps surprisingly, we find that under certain regimes (e.g., small tuple sizes, high k), it is beneficial for clients to simply download the entire collection instead of using Pung’s multi-retrieval.”
Pung throughput

The graph shows the throughput (messages/minute) for different numbers of clients (sending a single message) for three systems: Dissent, Pung, and Vuvuzela. The x-axis represents the number of clients, ranging from 64 to 131K, and the y-axis represents the throughput in messages per minute, ranging from $10^0$ to $10^6$.

- For 64 clients, Dissent has the lowest throughput, followed by Pung and then Vuvuzela.
- As the number of clients increases to 32K, the throughput for all systems increases, with Vuvuzela still leading.
- At 65K clients, the throughput continues to rise for all systems, with Vuvuzela maintaining its lead.
- At 131K clients, Vuvuzela still has the highest throughput, followed by Pung and then Dissent.

The graph indicates that Vuvuzela is the most efficient in terms of throughput for the given scenario.
Towards Vuvuzela

Alice

\[ E_{pk1}(E_{pk2}(E_{pk3}(E_{pkBob}(M)))) \]

Bob

What is protected? What is leaked?
- Set of users who sent a message
- Set of users who received a message

Have all users always send a message (can be dummy)

Don’t reveal recipient in final plaintext. All users download all final ciphertexts. Trial decrypt
Vuvuzela

- Weaken security goal to differential privacy (more in a second)
- Have final node store message at deaddrop $F_K(Alice, Bob)$
- Every round is both read and write through mixnet for all users
  - No communication, send dummy message to random tag
  - Messages sent to same deaddrop sent back through via reverse onion encryption
- Must use counters to avoid repeat use of tag $F_K(Alice, Bob)$
**Vuvuzela**

\[ E_{pk1}(E_{pk2}(E_{pk3}(F_K(Alice, Bob) \ || \ E_{pkBob}(M)))) \]

What still leaks if we stop here?

If Alice, Bob communicating, must be double access to a tag

Adversary drops all others’ communications and sees if there’s a double access to any tag... Confirms Alice, Bob communicating
Vuvuzela

$$E_{pk1}(E_{pk2}(E_{pk3}(F_K(Alice, Bob) || E_{pkBob}(M))))$$

Servers carefully add dummy messages to get differential privacy for leakage (# of double accesses and # of single accesses)

Whether Alice, Bob communicating or not gives rise to approximately same distribution of double deaddrop accesses
Vuvuzela DP goal

Let $M$ be algorithm that adds noise to # of single accesses and # of double accesses. Then $M$ is $(\varepsilon, \delta)$-DP if

$$\Pr[M(x) \in S] \leq e^\varepsilon \Pr[M(y) \in S] + \delta$$

Thm: Amount of noise scales with $\sqrt{k}$ for $k = \#$ of rounds, independent of number of users

They target $(\ln 2, 10^{-4})$-DP.
Fixes the number of rounds one can get this level of DP for Bounds degrade if one goes beyond this number of rounds
Vuvuzela DP goal

Let $M$ be algorithm that adds noise to # of single accesses and # of double accesses. Then $M$ is $(\varepsilon, \delta)$-DP if

$$\Pr[M(x) \in S] \leq e^{\varepsilon} \Pr[M(y) \in S] + \delta$$
Doesn’t count “dialing protocol” for telling someone that you want to talk. Double all latencies. Also does not include variance due to noise (!!)
Is it practical?

- What does one do as # of rounds increases?
  - Nothing... system doesn’t provide meaningful formal security guarantees
- Latency fundamentally slow (must wait for all messages from all participants). See follow-up work [Tyagi et al. 2017]
- Expensive: $30k a month to run 3 servers

Broader issues of all recent systems:
- Users must get other’s keys out-of-band
- Clients must always participate when online
  - Huge waste of bandwidth!
Security levels

- No-trust model
- All-but-one model
- “Plausible deniability” model (differential privacy)

None prevent *intersection attacks*: can’t prevent leakage when whether client is online or not is correlated with who they are talking to.