Searching on Encrypted Data and Timing Attacks

Dawn Song
dawnsong@cs.berkeley.edu

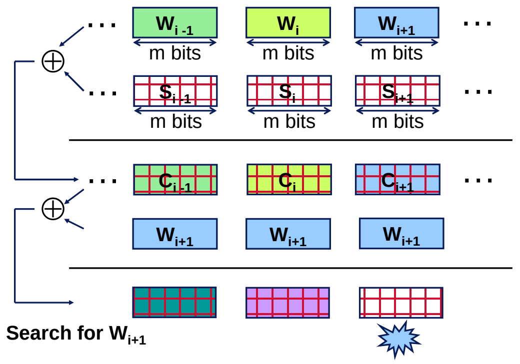
Motivating Example: Equality Search on Encrypted Data

- **Searching encrypted e-mails on servers**
- **Searching encrypted files on servers**
- **Searching in encrypted databases**

Desired Properties

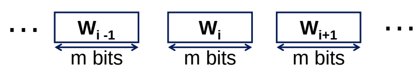
- **Word search is provably secure**
  - **Provable encryption properties**
  - **Server cannot search for arbitrary words**
  - **Does not leak information about other words**
  - **Does not reveal query word**
- **Efficiency**
  - **Low computation overhead**
  - **Low space and communication overhead**
  - **Low management overhead**

The Key Idea



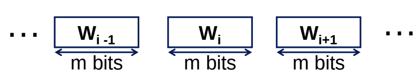
Setup and Notations

- **Document: sequence of fixed length words**



Setup and Notations

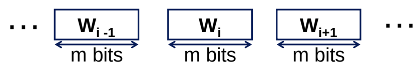
- **Document: sequence of fixed length words**



- **L_0, L_1, L_2, …**
  – **sequence of pseudorandom n-bit blocks**

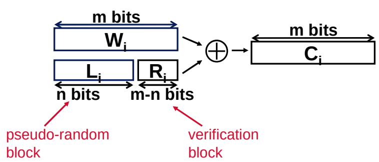
Setup and Notations

- **Document: sequence of fixed length words**



- **$L_0, L_1, L_2, \ldots$**
  - **sequence of pseudorandom n-bit blocks**
- **Pseudorandom Function F_K**
  - **maps n bits to (m-n) bits**

Basic Scheme (Encryption)

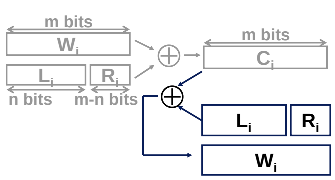


We xor the word W_i with

$L_i \leftarrow$ pseudorandom bits

$R_i \leftarrow F_K(L_i)$

Basic Scheme (Decryption)

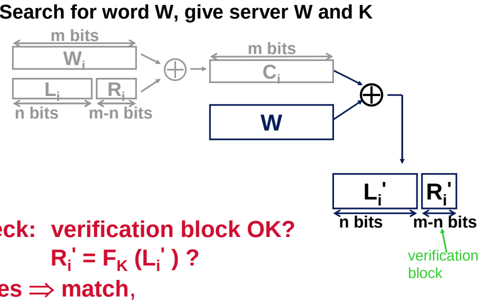


$L_i \leftarrow$ pseudorandom bits

$R_i \leftarrow F_K(L_i)$

Basic Scheme (Searches)

Search for word W, give server W and K



Check: verification block OK?

$R_i' = F_K(L_i')$?

Yes ⇒ match,

false positive rate = $1 / 2^{m-n}$

Controlled Searches and Query Isolation

- **To keep server from searching for arbitrary words & To avoid leaking information about other words**
- **In encryption:**

  Replace

  $R_i \leftarrow F_K(L_i)$

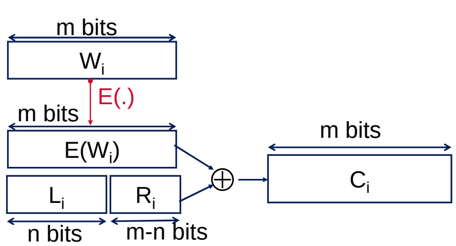
  with

  $R_i \leftarrow F_{K_i}(L_i)$, where $K_i = F'_K(W_i)$
- **To search for word W:**

  Reveal

  $K_W = F'_K(W)$
- **Enhancements:**
  - **Check only for “word occurs at least once” in document**
  - **Check only for “word occurs at least N times” in document**

Hidden Queries

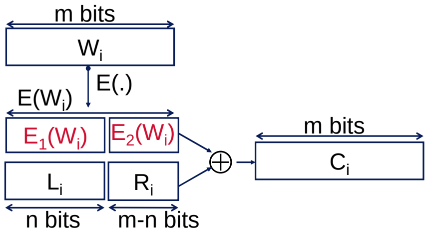


$L_i \leftarrow$ pseudorandom bits

$R_i \leftarrow F_{K_i}(L_i)$

where $K_i = F'_K(E(W_i))$

Final Scheme (Encryption)



$L_i \leftarrow$ ***pseudorandom bits***

$R_i \leftarrow F_{K_i}(L_i)$

where $K_i = F'_K(E_1(W_i))$

Summary for Keyword Search on Encrypted Data (Symmetric Key Case)

- **Provable security**
  - **Provable secrecy**
  - **Controlled search**
  - **Query isolation**
  - **Hidden queries**
- **Simple and efficient**
  - **O(length of document) stream cipher, block cipher and MAC operations for encryption/decryption**
  - **O(length of document) MAC operations for search**
  - **Almost no space and communication overhead**
  - **Easy to add documents**
  - **Convenient key management : user needs only one master key**

Administrative Matters

- **Out of town starting Wed**
- **My office hour this week will be 5pm Tue**
- **John will give a guest lecture on Wed**
- **Rusty and Todd will do midterm review next Mon**

Midterm Scope (I)

- Symmetric key encryption
  - Concept
  - One-time pad
  - Block cipher modes: how they work
- Public key encryption
  - Concept
  - How does RSA encryption/decryption work?
  - How does ElGamal encryption/decryption work?
- Hash functions
  - Concept of one-way, pre-image resistance, 2nd pre-image resistance, collision resistance
- Message authentication
  - Concept
    - E.g., what's the difference between the concept of encryption and message authentication?

Midterm Scope (II)

- Digital signatures
  - Concept
    - E.g., what's the difference between digital signatures & MACs
  - One-time signature
  - ElGamal signature
  - RSA signature
- Secret sharing
  - Concept
  - Threshold secret sharing schemes
- Zero-knowledge proofs
  - Concept
  - ZKP of square roots and other graph-based examples

Midterm Scope (III)

- Authentication and key exchange protocols
  - Identify potential attacks
  - Do not need to know how exactly every message works
- Random number generator
  - How to generate random numbers in practice
  - Which sources are potentially good/bad sources of randomness

Side-Channel Attacks on Crypto

- **A different attacker model**
  - **Side-channel attacks on Crypto**
- **Example: RSA in OpenSSL was vulnerable to timing attack:**
  - **Attacker can extract RSA private key by measuring web server response time**
- **Exploiting OpenSSL's timing vulnerability:**
  - **One process can extract keys from another.**
  - **Extract web server key remotely.**
    - **Our attack works across Stanford campus.**

Background: RSA Decryption

- **RSA decryption: g^d mod N = m**
  - **d is private decryption exponent, N is public modulus**
- **Chinese remaindering (CRT) uses factors directly. N=pq, and d1 and d2 are pre-computed from d:**
  1. **m1 = g^{d1} mod q**
  2. **m2 = g^{d2} mod p**
  3. **combine m1 and m2 to yield m (mod N)**
- **Goal: learn factors of N.**
  - **Kocher's [K'96] attack fails when CRT is used.**

RSA Decryption Time Variance

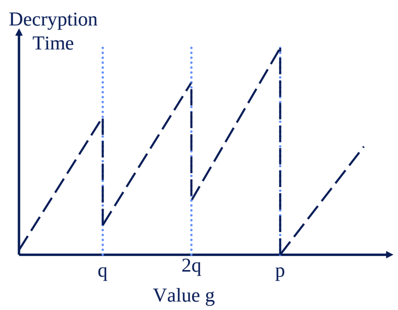
- **Causes for decryption time variation:**
  - **Which multiplication algorithm is used.**
    - **OpenSSL uses both basic mult. and Karatsuba mult.**
  - **Number of steps during a modular reduction**
    - **modular reduction goal: given u, compute u mod q**
    - **Occasional extra steps in OpenSSL's reduction alg.**
- **There are MANY:**
  - **multiplications by input g**
  - **modular reductions by factor q (and p)**

Reduction Timing Dependency

- **Modular reduction:** **given u, compute u mod q.**
  - **OpenSSL uses Montgomery reductions [M'85] .**
- **Time variance in Montgomery reduction:**
  - **One extra step at end of reduction algorithm with probability**

$$\Pr[\text{extra step}] \approx \frac{(g \bmod q)}{2q} \quad \text{[S'00]}$$

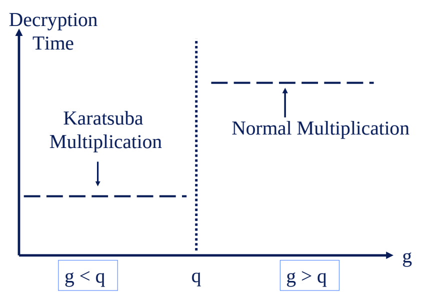
$\Pr[\text{extra step}] \approx \frac{(g \bmod q)}{2q}$



Multiplication Timing Dependency

- **Two algorithms in OpenSSL:**
  - **Karatsuba (fast): Multiplying two numbers of equal length**
  - **Normal (slow): Multiplying two numbers of different length**
- **To calc x·g mod q OpenSSL does:**
  - **When x is the same length as (g mod q), use Karatsuba mult.**
  - **Otherwise, use Normal mult.**

Multiplication Summary



Data Dependency Summary

- **Decryption value g < q**
  - **Montgomery effect: longer decryption time**
  - **Multiplication effect: shorter decryption time**
- **Decryption value g > q**
  - **Montgomery effect: shorter decryption time**
  - **Multiplication effect: longer decryption time**

Opposite effects! But one will always dominate

Timing Attack

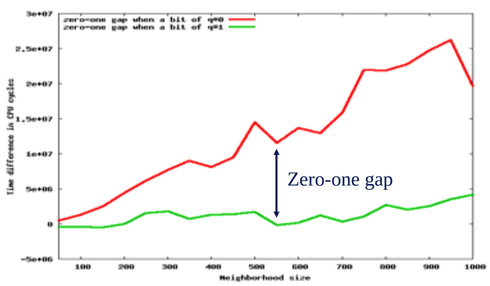
High Level Attack:

1) **Suppose g=q for the top i-1 bits, and 0 elsewhere.**
2) **g_{hi} = g, but with the i^{th} bit 1. Then $g < g_{hi}$**
   Goal: decide if $g<q<g_{hi}$ or $g<g_{hi}<q$
3) **Sample decryption time for g and g_{hi}:**
   t_1 = DecryptTime(g)
   t_2 = DecryptTime(g_{hi})
   (large vs. small creates 0-1 gap)
4) **If $|t_1 - t_2|$ is large** ⇒ g and g_{hi} straddle q ⇒ **bit i is 0 ($g < q < g_{hi}$)**
   else ⇒ don't straddle q ⇒ **bit i is 1 ($g < g_{hi} < q$)**

Timing Attack Details

- We know what is "large" and "small" from attack on previous bits.
- Decrypting just g does not work because of sliding windows
  - Decrypt a neighborhood of values near g
  - Will increase diff. between large and small values ⇒ larger 0-1 gap
- Only need to recover q/2 bits of q [C'97]
- Attack requires only 2 hours, about 1.4 million queries

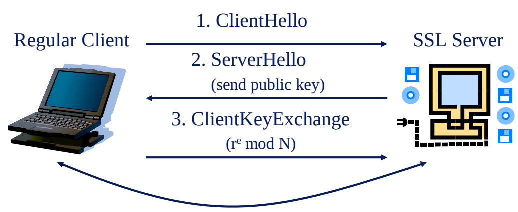
The Zero-One Gap



How does this work with SSL?

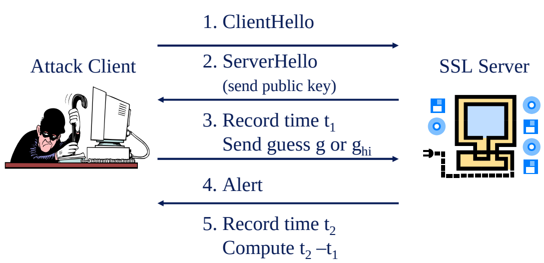
How do we get the server to decrypt our g?

Normal SSL Decryption



Result: Encrypted with computed shared master secret

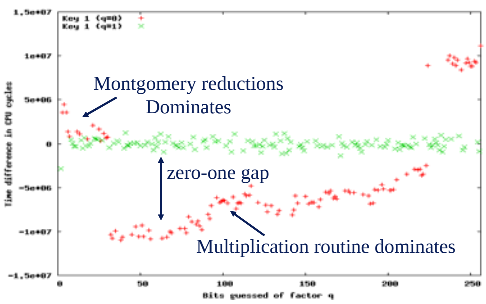
Attack SSL Decryption



Attack requires accurate clock

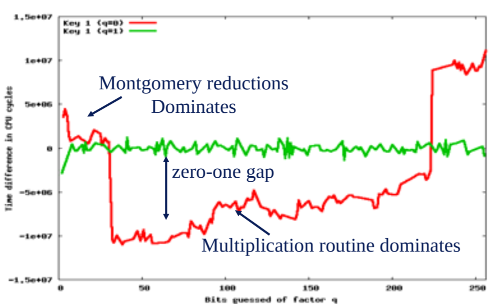
- **Attack measures 0.05% time difference between g and g_{hi}**
  - **Only 0.001 seconds on a P4**
- **We use the CPU cycle counter as fine-resolution clock**
  - **"rdtsc" instruction on Intel**
  - **"%tick" register on UltraSparc**

Attack extract RSA private key



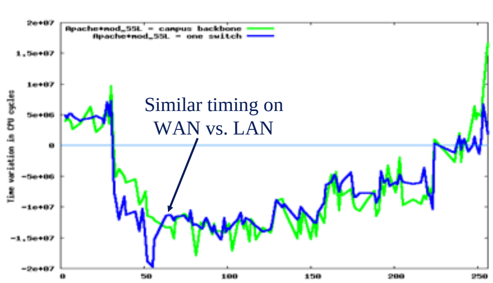
34

Attack extract RSA private key



35

Attack works on the network



36

Attack Summary

- **Attack successful, even on a WAN**
- **Attack requires only 350,000 – 1,400,000 decryption queries.**
- **Attack requires only 2 hours.**

37

Defenses

Good: Use RSA blinding

BAD: Require statically all decryptions to take the same time

BAD: Use dynamic methods to make all decryptions take the same time

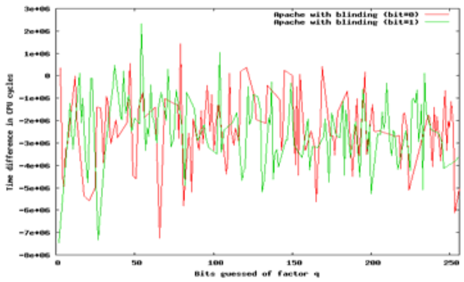
38

RSA Blinding

- **Decrypt random number related to g:**
  1. **Compute x' = g*r^e mod N, r is random**
  2. **Decrypt x' = m'**
  3. **Calculate m = m'/r mod N**
- **Since r is random, the decryption time should be random**
- **2-10% performance penalty**

39

Blinding Works!



Conclusion

- Side-channel attacks
  - Different attacker model can break security
- Crypto libraries should defend against side-channel attacks

Conclusion

- We developed a timing attack based on multiplication and reduction timings
- Attack works against real OpenSSL-based servers on regular PC's.
- Lesson: Crypto libraries should always defend against timing attacks.

Conclusion

- Ecash
- Search/computation on encrypted data

Conjunctive Equality Test on Encrypted Data (Public Key Case)

- Example:
  - Check whether an encrypted file contain every keyword in a set
  - Subset, range query

Motivating example II: Multi-dimensional Range Query on Encrypted Data

- **Network audit logs**
  - **Encrypted**
  - **An auditor may only be able to decrypt entries satisfying certain predicates**
    - » **E.g., p1< port p < p2, timestamp t > t1, source address a with prefix a1**

Traditional Encryption

- **Semantic security**
  - **Given $E_k\{b\}$, difficult to guess b=0 or 1**
- **Search on encrypted data**
  - **Semantic security not suitable, by definition**
  - **Instead, encryption with search capability**
    - » **Predicate encryption**

Predicate Encryption (Symmetric Key Case)

- **Let $\Phi = \{P_1, \ldots, P_n\}$ be a set of predicates over Σ.**

  $P_i : \Sigma \rightarrow \{0,1\}$ [e.g: $P_j(S) = 1 \Leftrightarrow S \geq j$]
- **A Φ-query system consists of 4 algorithms:**
  - **Setup (λ): outputs SK**
  - **Encrypt (SK, S) $\rightarrow$ Ciphertext C ($S \in \Sigma$)**
  - **GenCapability (SK, <P>) $\rightarrow$ Capability T_P ($P \in \Phi$)**
  - **Query (T_P, C) $\rightarrow$ Output** P(S)
  - **(Can allow message decryption on "hit" when P(S)=1)**

Predicate Encryption (Public Key Case)

- Let $\Phi = \{P_1, \ldots, P_n\}$ be a set of predicates over Σ.

  $P_i : \Sigma \rightarrow \{0,1\}$ [e.g: $P_j(S) = 1 \Leftrightarrow S \geq j$]

- A Φ-query system consists of 4 algorithms:
  - Setup (λ): outputs PK and SK
  - Encrypt (PK, S) $\rightarrow$ Ciphertext C ($S \in \Sigma$)
  - GenCapability (SK, <P>) $\rightarrow$ Capability T_P ($P \in \Phi$)
  - Query (T_P, C) $\rightarrow$ Output P(S)
  - (Can allow message decryption on "hit" when P(S)=1)

49

Security

- Learn nothing more than the given search capabilities

- Why do we need to construct the search capability? What if the encryption algorithm allows anyone to search for anything?

50

State-of-the-Art (I)

- **Equality test:**
  - Symmetric-key Case
    - Goldreich, Ostrovsky, [JACM 1996]
    - Song, Wagner, Perrig, [S&P 2000]
  - Public-key case
    - Boneh, Crescenzo, Ostrovsky, Persiano, [Eurocrypt 2004]

$$f_a(X) = \begin{cases} 1 & X = a \\ 0 & o.w. \end{cases}$$

51

State-of-the-Art (II)

- **Multi-dimensional range queries:** $X = (x_1, x_2, \ldots, x_n)$ [SBCSP06]

$$f_{a_1,a_2,b_1,b_2}(X) = \begin{cases} 1 & (x_1 \in [a_1, a_2]) \wedge (x_3 \in [b_1, b_2]) \\ 0 & o.w. \end{cases}$$

(IP ∈ 128.2.*.*) ∧ (port ∈ [1000, 2000])

(IP ∈ 128.2.*.*) ∧ (port = 1434)

- **Core technique: conjunctive queries [SBCSP06,BW06]**

$$f_{a,b}(X) = \begin{cases} 1 & (x_1 = a) \wedge (x_3 = b) \\ 0 & o.w. \end{cases}$$

Equality Test on Encrypted Data (Symmetric Key Case)

- **Example:**
  - **Check whether an encrypted file contain a keyword**
  - **App: keyword search on encrypted emails**