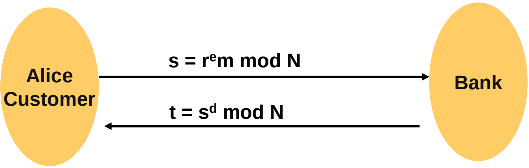
Random Number Generation and Electronic Cash

Dawn Song

dawnsong@cs.berkeley.edu

Building Block: Blind Signatures

- **Blind signature: achieve anonymity**
  - **How can Alice get a signature from the Bank without the Bank knowing what message is being signed?**
- **Protocol:**
  - **generating blind signature on message m in RSA setting**
  - **Bank's private key (d, p, q), public key (e, N)**



$s = r^e m \bmod N$

$t = s^d \bmod N$

- **Alice computes $t/r \bmod N = m^d \bmod N$**

Ecash Using Blind Signature

- **How to use blind signature to build ecash?**
- **A valid $1 bill is a pair (x,y), where $y = \text{hash}(x)^d \bmod N$, hash() is one-way function**
- **How does the ecash protocol work?**
- **Why do we need hash()?**
- **How to prevent double spending?**
- **What to do for different denominations?**
  - **Nickles, dimes, dollars**

Other Methods for Ecash

- Use zero-knowledge proofs (out of scope)
  - More building blocks of ZKP
  - Support many properties
    - Identifying double spenders

Administrative Matters

- Next class: talk about midterm scope

Untrusted Storage

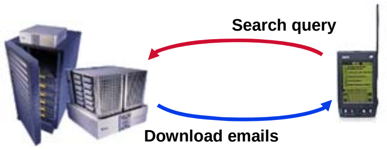
- User’s sensitive data often stored in untrusted storage
  - Third party storage/out-sourced storage
  - Mis-configuration causes information leakage
  - Attacker hacks into system
  - Insider attack
- Need to encrypt data to protect privacy
- Yet, need to perform certain operations on data for functionality

Operations on Encrypted Data

- What kind of operations needed on encrypted data?
- What are your favorite applications that you would like to enable on encrypted data?
- Here we focus on searching on encrypted data

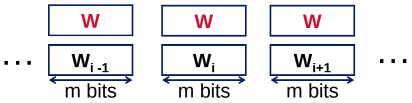
Motivating Example: Equality Search on Encrypted Data

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



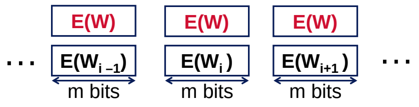
Sequential Scan

Search for W



Sequential Scan – with encryption

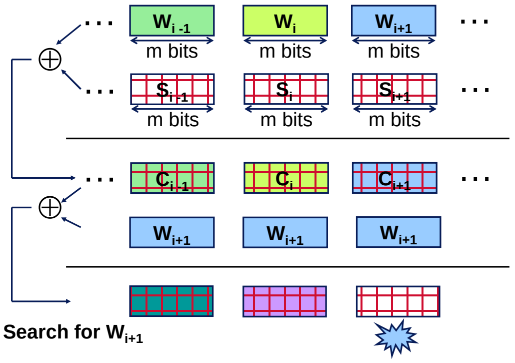
Search for W



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**

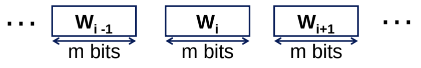
The Key Idea



Search for W_{i+1}

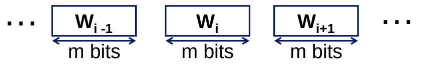
Setup and Notations

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



Setup and Notations

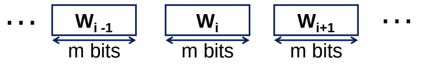
- **Document: sequence of fixed length words**



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

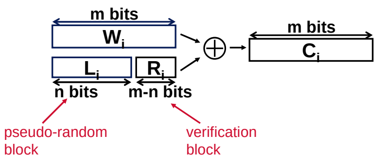
Setup and Notations

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



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

Basic Scheme (Encryption)

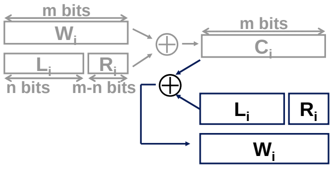


We xor the word W_i with

$L_i \leftarrow$ *pseudorandom bits*

$R_i \leftarrow F_K(L_i)$

Basic Scheme (Decryption)

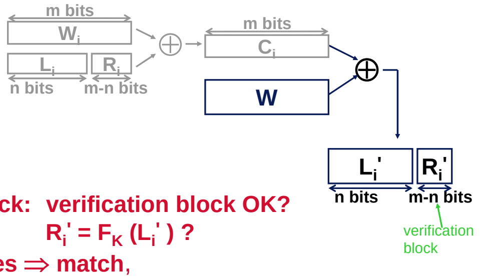


$L_i \leftarrow$ *pseudorandom bits*

$R_i \leftarrow F_K(L_i)$

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

  $R_i \leftarrow F_K(L_i)$

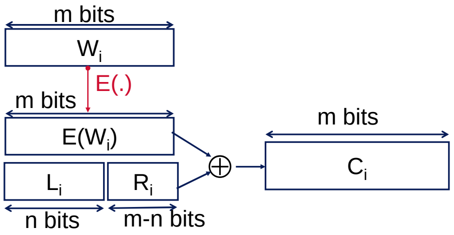
  with

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

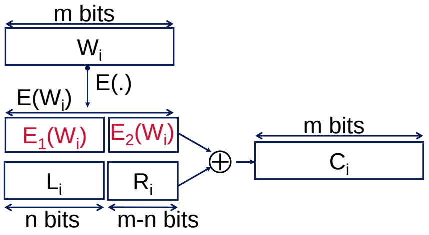


$L_i \leftarrow$ *pseudorandom bits*

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

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

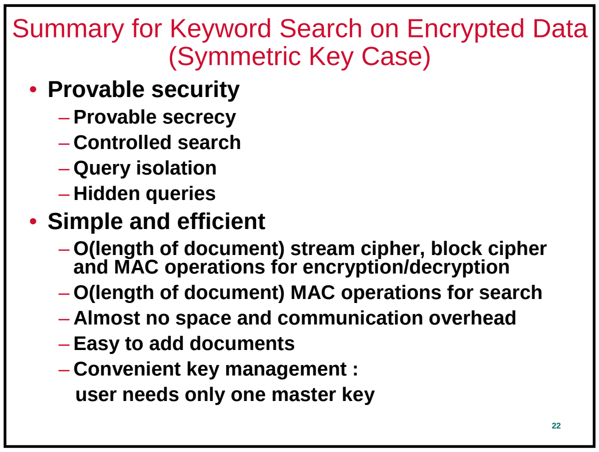
Final Scheme (Encryption)



$L_i \leftarrow$ *pseudorandom bits*

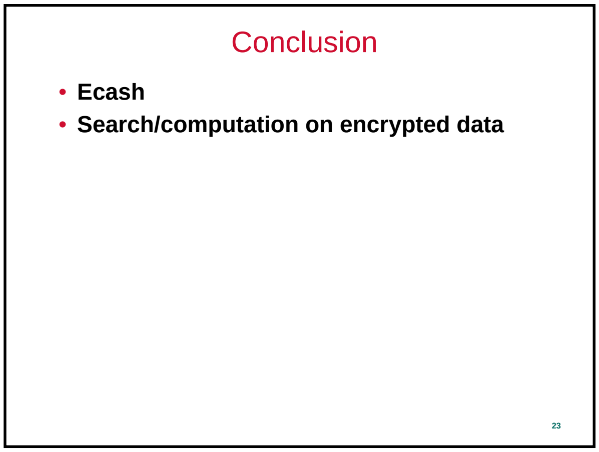
$R_i \leftarrow F_{K_i}(L_i)$

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



Conclusion

- Ecash
- Search/computation on encrypted data