Introduction to Cryptography (cont.)

Daniel Halperin
Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Updates Oct. 12th

• **Coffee/tea** signup sheet posted (optional)
  • Next is Friday @11 am. Meet in CSE Atrium

• **Lab 1** due next Friday (10/21) @5pm

• TA office hours Friday before class (CSE 002)

• My office hours today after class (CSE 210)

• **Reading:** over the next few days, Crypto chapters (Ch. 12--15, ~50 pages) in Daswani et al.
Symmetric Setting

Both communicating parties have access to a shared random string $K$, called the key.
Asymmetric Setting

Each party creates a public key \( pk \) and a secret key \( sk \).
Where do keys come from?

(http://4.bp.blogspot.com/_8MUCc1TyEQ0/SV1WcICKXul/AAAAAAAARo/Vh5Jr929oT4/s1600-h/stork)
“Random” Numbers

Pseudorandom Number Generators (PRNGs)

PRNG

R₁, R₂, R₃, R₄, R₅, ...

Machine State

User Input

Adversary

Alice
Getting keys: PBKDF

Password-based Key Derivation Functions

Password \rightarrow \text{PBKDF} \rightarrow K

(K key check value)
Getting keys: CAs

Each party creates a public key \( pk \) and a secret key \( sk \).

/Public keys signed by a trusted third party: a certificate authority./
Getting keys: CAs

Each party creates a public key $pk$ and a secret key $sk$.

(Public keys signed by a trusted third party: a certificate authority.)
Getting keys: CAs

Each party creates a public key $pk$ and a secret key $sk$.

(Public keys signed by a trusted third party: a certificate authority.)
Getting keys: CAs

Each party creates a public key $pk$ and a secret key $sk$.

(Public keys signed by a trusted third party: a certificate authority.)
Getting keys: CAs

Each party creates a public key $pk$ and a secret key $sk$.

(Public keys signed by a trusted third party: a certificate authority.)
Getting keys: Key exchange

Key exchange protocols: A tool for establishing a shared symmetric key from public keys
Getting keys: Key exchange

Key exchange protocols: A tool for establishing a shared symmetric key from public keys
One-way Communications

PGP is a good example

Message encrypted under Bob’s public key
Interactive Communications

In many cases, it’s probably a good idea to just use a standard protocol/system like SSH, SSL/TLS, etc...

Let’s talk securely; here are the algorithms I understand

I choose these algorithms; start key exchange

Continue key exchange

Communicate using exchanged key
Let’s Dive a Bit Deeper
One-way Communications

*(Informal example; ignoring, e.g., signatures)*
One-way Communications

*(Informal example; ignoring, e.g., signatures)*

1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
One-way Communications

*(Informal example; ignoring, e.g., signatures)*

1. Alice gets Bob’s public key; Alice *verifies* Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
One-way Communications

*Informal* example; ignoring, e.g., signatures

1. Alice gets Bob’s public key; Alice *verifies* Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys $K_1$ and $K_2$
3. Alice encrypts the message $M$ the key $K_1$; call result $C$
One-way Communications

(*Informal* example; ignoring, e.g., signatures)

1. Alice gets Bob’s public key; Alice *verifies* Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
3. Alice encrypts the message M the key K1; call result C
4. Alice authenticates (MACs) C with key K2; call the result T
One-way Communications

(Informal example; ignoring, e.g., signatures)

1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys $K_1$ and $K_2$
3. Alice encrypts the message $M$ the key $K_1$; call result $C$
4. Alice authenticates (MACs) $C$ with key $K_2$; call the result $T$
5. Alice encrypts $K_1$ and $K_2$ with Bob’s public key; call the result $D$
One-way Communications

(Informal example; ignoring, e.g., signatures)

1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
3. Alice encrypts the message M the key K1; call result C
4. Alice authenticates (MACs) C with key K2; call the result T
5. Alice encrypts K1 and K2 with Bob’s public key; call the result D
6. Send D, C, T
One-way Communications

*(Informal example; ignoring, e.g., signatures)*

1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
3. Alice encrypts the message M the key K1; call result C
4. Alice authenticates (MACs) C with key K2; call the result T
5. Alice encrypts K1 and K2 with Bob’s public key; call the result D

6. Send D, C, T

(Assume Bob’s private key is encrypted on Bob’s disk.)
One-way Communications

*(Informal example; ignoring, e.g., signatures)*

1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
3. Alice encrypts the message M the key K1; call result C
4. Alice authenticates (MACs) C with key K2; call the result T
5. Alice encrypts K1 and K2 with Bob’s public key; call the result D
6. Send D, C, T
7. Bob takes his password to derive key K3
One-way Communications

*(Informal example; ignoring, e.g., signatures)*

1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys $K_1$ and $K_2$
3. Alice encrypts the message $M$ with the key $K_1$; call result $C$
4. Alice authenticates (MACs) $C$ with key $K_2$; call the result $T$
5. Alice encrypts $K_1$ and $K_2$ with Bob’s public key; call the result $D$

6. Send $D, C, T$

   (Assume Bob’s private key is encrypted on Bob’s disk.)

7. Bob takes his password to derive key $K_3$
8. Bob decrypts his private key with key $K_3$
One-way Communications

(Informal example; ignoring, e.g., signatures)
1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
3. Alice encrypts the message M the key K1; call result C
4. Alice authenticates (MACs) C with key K2; call the result T
5. Alice encrypts K1 and K2 with Bob’s public key; call the result D

6. Send D, C, T

(Assume Bob’s private key is encrypted on Bob’s disk.)
7. Bob takes his password to derive key K3
8. Bob decrypts his private key with key K3
9. Bob uses private key to decrypt K1 and K2
One-way Communications

*(Informal example; ignoring, e.g., signatures)*

1. Alice gets Bob’s public key; Alice *verifies* Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys $K_1$ and $K_2$
3. Alice encrypts the message $M$ the key $K_1$; call result $C$
4. Alice authenticates (MACs) $C$ with key $K_2$; call the result $T$
5. Alice encrypts $K_1$ and $K_2$ with Bob’s public key; call the result $D$

6. Send $D, C, T$

(Assume Bob’s private key is encrypted on Bob’s disk.)

7. Bob takes his password to derive key $K_3$
8. Bob decrypts his private key with key $K_3$
9. Bob uses private key to decrypt $K_1$ and $K_2$
10. Bob uses $K_2$ to verify MAC tag $T$
One-way Communications

*(Informal example; ignoring, e.g., signatures)*

1. Alice gets Bob’s public key; Alice verifies Bob’s public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
3. Alice encrypts the message M the key K1; call result C
4. Alice authenticates (MACs) C with key K2; call the result T
5. Alice encrypts K1 and K2 with Bob’s public key; call the result D

6. Send D, C, T

(Assume Bob’s private key is encrypted on Bob’s disk.)

7. Bob takes his password to derive key K3
8. Bob decrypts his private key with key K3
9. Bob uses private key to decrypt K1 and K2
10. Bob uses K2 to verify MAC tag T
11. Bob uses K1 to decrypt C
Interactive Communications

(*Informal* example; details omitted)
Interactive Communications

(*Informal example; details omitted*)

1. Alice and Bob exchange public keys and certificates
Interactive Communications

*(Informal example; details omitted)*

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA’s public keys to verify certificates and each other’s public keys
Interactive Communications

*(Informal example; details omitted)*

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA’s public keys to verify certificates and each other’s public keys
3. Alice and Bob take their passwords and derive symmetric keys
Interactive Communications

(Informal example; details omitted)

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA’s public keys to verify certificates and each other’s public keys
3. Alice and Bob take their passwords and derive symmetric keys
   4. Alice and Bob use those symmetric keys to decrypt and recover their asymmetric private keys.
Interactive Communications

*(Informal example; details omitted)*

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA’s public keys to verify certificates and each other’s public keys
3. Alice and Bob take their passwords and derive symmetric keys
4. Alice and Bob use those symmetric keys to decrypt and recover their asymmetric private keys.
5. Alice and Bob use their asymmetric private keys and a key exchange algorithm to derive a shared symmetric key
Interactive Communications

(Informal example; details omitted)

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA’s public keys to verify certificates and each other’s public keys
3. Alice and Bob take their passwords and derive symmetric keys
4. Alice and Bob use those symmetric keys to decrypt and recover their asymmetric private keys.
5. Alice and Bob use their asymmetric private keys and a key exchange algorithm to derive a shared symmetric key
   (They key exchange process will require Alice and Bob to generate new pseudorandom numbers)
Interactive Communications

(*Informal* example; details omitted)

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA’s public keys to verify certificates and each other’s public keys
3. Alice and Bob take their passwords and derive symmetric keys
4. Alice and Bob use those symmetric keys to decrypt and recover their asymmetric private keys.
5. Alice and Bob use their asymmetric private keys and a *key exchange* algorithm to derive a shared symmetric key
   (They key exchange process will require Alice and Bob to generate new pseudorandom numbers)
6. Alice and Bob use shared symmetric key to encrypt and authenticate messages
Interactive Communications

(Informal example; details omitted)

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA’s public keys to verify certificates and each other’s public keys
3. Alice and Bob take their passwords and derive symmetric keys
4. Alice and Bob use those symmetric keys to decrypt and recover their asymmetric private keys.
5. Alice and Bob use their asymmetric private keys and a key exchange algorithm to derive a shared symmetric key
   (They key exchange process will require Alice and Bob to generate new pseudorandom numbers)
6. Alice and Bob use shared symmetric key to encrypt and authenticate messages
   (Last step will probably also use random numbers; will need to rekey regularly; may need to avoid replay attacks,...)
What cryptosystems have you heard of? (Past or present)
History

- Substitution Ciphers
  - Caesar Cipher
- Transposition Ciphers
- Codebooks
- Machines

- Recommended Reading: The Codebreakers by David Kahn and The Code Book by Simon Singh.
  - Military uses
  - Rumrunners
  - ....
Classic Encryption

• Goal: To communicate a secret message
• Start with an algorithm
• Caesar cipher (substitution cipher):

   ABCDEFGHIJKLMNOPQRSTUVWXYZ
   GHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOP

Wednesday, October 12, 11
Then add a secret key

• Both parties know that the secret word is “victory”:

  ABCDEFGHIJKLMNOPQRSTUVWXYZ

  **VICTORY**ABDEFGHJKLMNOPQRSTUVWXYZ

• “state of the art” for thousands of years
Kerckhoff’s Principle

- Security of a cryptographic object should depend **only** on the secrecy of the secret (private) key.

- Security should not depend on the secrecy of the algorithm itself.

- Why?