Introduction to cryptographic protocols
models, proofs, and attacks

Karthikeyan Bhargavan

INRIA
karthikeyan.bhargavan@inria.fr
http://prosecco.inria.fr/personal/karthik

September 2013

(Based on slides by Stéphanie Delaune, Bruno Blanchet, and others)
Cryptographic protocols

1. Cryptography

2. Protocol
Cryptographic protocols

- Cryptography
  - The study of mathematical techniques related to aspects of information security such as confidentiality and data integrity

- Protocol
Cryptography

- The study of mathematical techniques related to aspects of information security such as confidentiality and data integrity

- Crypto Primitives: algorithms for encryption, signature, hashing, ...
Cryptographic protocols

- Cryptography
  - *The study of mathematical techniques related to aspects of information security such as confidentiality and data integrity*
  - **Crypto Primitives**: algorithms for encryption, signature, hashing, . . .
  - **Examples**: RSA, AES, RC4 (encryption), RSA, DSA (signature), SHA-1, MD5 (hashing), HMAC, CMAC (MAC), . . .

- Protocol
Cryptography

The study of mathematical techniques related to aspects of information security such as confidentiality and data integrity

Crypto Primitives: algorithms for encryption, signature, hashing, ... 
Examples: RSA, AES, RC4 (encryption), RSA, DSA (signature), SHA-1, MD5 (hashing), HMAC, CMAC (MAC), ...

Protocol

A set of rules governing the transmission and storage of data that is exchanged between computers.
Cryptography

- The study of mathematical techniques related to aspects of information security such as confidentiality and data integrity

- Crypto Primitives: algorithms for encryption, signature, hashing, ...
- Examples: RSA, AES, RC4 (encryption), RSA, DSA (signature), SHA-1, MD5 (hashing), HMAC, CMAC (MAC), ...

Protocol

- A set of rules governing the transmission and storage of data that is exchanged between computers.
- Examples: TCP/IP, GSM, Network File System, Cloud Storage
Cryptographic protocols

- **Cryptographic protocol**: A set of rules for the exchange of data between multiple principals that uses cryptography to achieve security goals against a threat model.
  - **Principal**: a protocol participant, typically human or computer
  - **Security Goal**: the confidentiality or integrity of a data item, or the authentication of a principal
  - **Threat Model**: the capabilities of the attacker

- **Examples**
  - Communications protocols: TLS, IPsec, SSH, WPA
  - Tamper-proof hardware: Smartcard, Navigo, SIM card
  - Privacy preserving applications: BitCoin, Electronic Voting
Example: Online Banking

Secure connection to bank’s website

Nobody other than the bank can read what I type (confidentiality)

My secret login information

Nobody other than me can access my account page (authentication)

Goal: Prevent unauthorized access to data even if an unknown attacker controls the network and some other bank clients.
Example: Online Banking

- **Cryptographic Protocol**: TLS (HTTPS)

- **Cryptographic Protocol**: Password-based authentication
Example: Online Banking

- **Cryptographic Protocol**: TLS (HTTPS)
  - **Principals**: Web Browser, Bank Website

- **Cryptographic Protocol**: Password-based authentication
  - **Principals**: Bank Client, Bank Website

Security Goal: confidentiality and integrity of data (secure channel), server authentication

Threat Model: network attacker (malicious wireless access point), phishing website
Example: Online Banking

- **Cryptographic Protocol:** TLS (HTTPS)
  - **Principals:** Web Browser, Bank Website
  - **Security Goal:** confidentiality and integrity of data (secure channel), server authentication

- **Cryptographic Protocol:** Password-based authentication
Example: Online Banking

- **Cryptographic Protocol**: TLS (HTTPS)
  - **Principals**: Web Browser, Bank Website
  - **Security Goal**: confidentiality and integrity of data (secure channel), server authentication
  - **Threat Model**: network attacker (malicious wireless access point), phishing website

- **Cryptographic Protocol**: Password-based authentication

Karthikeyan Bhargavan (INRIA)
Introduction to cryptographic protocols
September 2013
Example: Online Banking

- **Cryptographic Protocol**: TLS (HTTPS)
  - **Principals**: Web Browser, Bank Website
  - **Security Goal**: confidentiality and integrity of data (secure channel), server authentication
  - **Threat Model**: network attacker (malicious wireless access point), phishing website

- **Cryptographic Protocol**: Password-based authentication
  - **Principals**: Bank Client, Bank Website
Example: Online Banking

- **Cryptographic Protocol**: TLS (HTTPS)
  - **Principals**: Web Browser, Bank Website
  - **Security Goal**: confidentiality and integrity of data (secure channel), server authentication
  - **Threat Model**: network attacker (malicious wireless access point), phishing website

- **Cryptographic Protocol**: Password-based authentication
  - **Principals**: Bank Client, Bank Website
  - **Security Goal**: Client authentication
Example: Online Banking

- **Cryptographic Protocol**: TLS (HTTPS)
  - **Principals**: Web Browser, Bank Website
  - **Security Goal**: confidentiality and integrity of data (secure channel), server authentication
  - **Threat Model**: network attacker (malicious wireless access point), phishing website

- **Cryptographic Protocol**: Password-based authentication
  - **Principals**: Bank Client, Bank Website
  - **Security Goal**: Client authentication
  - **Threat Model**: dishonest client
Example: Credit Card Payment (EMV)

- Cardholder Verification (PIN Entry)
- Online Transaction Authorization
Example: Credit Card Payment (EMV)

- Cardholder Verification (PIN Entry)
  - **Principals**: Client, Terminal, Credit Card
  - **Security Goal**: Client authentication
  - **Threat Model**: Stolen credit card

- Online Transaction Authorization
Example: Credit Card Payment (EMV)

- **Cardholder Verification (PIN Entry)**
  - **Principals**: Client, Terminal, Credit Card
  - **Security Goal**: Client authentication
  - **Threat Model**: Stolen credit card

- **Online Transaction Authorization**
  - **Principals**: Credit Card, Terminal, Bank
  - **Security Goal**: Transaction data integrity, Card authentication
  - **Threat Model**: Forged credit card, Tampered terminal
Cryptographic protocols are small security-critical components embedded within large distributed applications.

**Example:** TLS within a web browser.

The security of the system depends on their correctness.
Cryptographic protocols are small security-critical components embedded within large distributed applications.

**Example**: TLS within a web browser

The security of the system depends on their correctness.

Still, a long history of attacks

- on academic protocols: see the SPORE repository
- on TLS (HTTPS): BEAST, CRIME, RC4
- on smartcards: YesCard, Side Channels

Why is it so hard to design secure protocols?
Cryptographic protocols are small security-critical components embedded within large distributed applications.

**Example**: TLS within a web browser.

The security of the system depends on their correctness.

Still, a long history of attacks:
- on academic protocols: see the SPORE repository
- on TLS (HTTPS): BEAST, CRIME, RC4
- on smartcards: YesCard, Side Channels

Why is it so hard to design secure protocols?
- Cryptographic guarantees are often misunderstood
- Rich threat models are difficult to reason about and to test
Cryptographic protocols are small security-critical components embedded within large distributed applications.

**Example**: TLS within a web browser.

The security of the system depends on their correctness.

Still, a long history of attacks:
- on academic protocols: see the SPORE repository
- on TLS (HTTPS): BEAST, CRIME, RC4
- on smartcards: YesCard, Side Channels

Why is it so hard to design secure protocols?
- Cryptographic guarantees are often misunderstood
- Rich threat models are difficult to reason about and to test

Our goal is to analyze the security of cryptographic protocols:
- develop mathematical proofs of correctness
- (or else find attacks)
Course Outline

- Today (K Bhargavan): Informal models and security analysis
- 3 weeks (K Bhargavan): Proofs in the symbolic (formal) model
- 4 weeks (D Pointcheval): Proofs in the computational model
- 4 weeks (B Blanchet): Tools and techniques for proof automation
- 4 weeks (K Bhargavan): Verifying protocol implementations
Informal Notation for Cryptographic Protocols
Example: Secure RPC
Formal and Computational Models
A Small Process Calculus
Example: Needham Schroeder Protocol
Informal Notation

- **Principals**: $A$ (alice), $B$ (bob), $C$ (charlie), 
- **Messages**: $m, n, o, \ldots$
Informal Notation

- **Principals**: A (alice), B (bob), C (charlie), ...
- **Messages**: $m, n, o, \ldots$
  - **Pairing**: $\langle m, n \rangle$
  - **Projection**: $\text{proj}_1(m)$, $\text{proj}_2(m)$
    - $\text{proj}_1(\langle m, n \rangle) = m$, $\text{proj}_2(\langle m, n \rangle) = n$

A protocol is informally specified as a sequence of messages exchanged between principals:
1. $A \rightarrow B$: $m_1$
2. $B \rightarrow C$: $m_2$
3. $C \rightarrow A$: $m_3$

Denotes the expected behaviour of a single run of the protocol. The goal of the attacker is to disrupt this behaviour!
Informal Notation

- **Principals**: A (alice), B (bob), C (charlie), ...
- **Messages**: $m, n, o, \ldots$
  - **Pairing**: $\langle m, n \rangle$
  - **Projection**: $\text{proj}_1(m), \text{proj}_2(m)$
    - $\text{proj}_1(\langle m, n \rangle) = m, \text{proj}_2(\langle m, n \rangle) = n$

A protocol is informally specified as a sequence of messages exchanged between principals:

1. $A \rightarrow B$: $m_1$
2. $B \rightarrow C$: $m_2$
3. $C \rightarrow A$: $m_3$
   ... 

- Denotes the expected behaviour of a single run of the protocol
- The goal of the attacker is to disrupt this behaviour!
Threat Model: *the network is the attacker*

- **Attacker Principals**: $E$ (eve), $M$ (mallory), $O$ (opponent), …
- Each message is sent over an insecure network: $A \rightarrow B : m$

These capabilities are enough to:
- Steal sensitive data (e.g. passwords, bank statements)
- Impersonate principals (e.g. bank clients and bank servers)
- Tamper with sensitive data (e.g. bank transactions)

To prevent such attacks, protocols rely on cryptography.
Threat Model: *the network is the attacker*

- **Attacker Principals**: $E$ (eve), $M$ (mallory), $O$ (opponent), …
- Each message is sent over an insecure network: $A \rightarrow B : m$
- **Interception**: The attacker can steal protocol messages: $A \rightarrow O(B) : m$

These capabilities are enough to:
- Steal sensitive data (e.g. passwords, bank statements)
- Impersonate principals (e.g. bank clients and bank servers)
- Tamper with sensitive data (e.g. bank transactions)

To prevent such attacks, protocols rely on cryptography
Threat Model: *the network is the attacker*

- **Attacker Principals**: $E$ (eve), $M$ (mallory), $O$ (opponent), ... 
- Each message is sent over an insecure network: $A \rightarrow B : m$
- **Interception**: The attacker can steal protocol messages: $A \rightarrow O(B) : m$
- **Injection**: The attacker can send protocol messages: $O(A) \rightarrow B : m$

These capabilities are enough to:

- Steal sensitive data (e.g. passwords, bank statements)
- Impersonate principals (e.g. bank clients and bank servers)
- Tamper with sensitive data (e.g. bank transactions)

To prevent such attacks, protocols rely on cryptography.
Threat Model: *the network is the attacker*

- **Attacker Principals**: $E$ (eve), $M$ (mallory), $O$ (opponent), ...  
- Each message is sent over an insecure network: $A \rightarrow B : m$
- **Interception**:
  - The attacker can steal protocol messages: $A \rightarrow O(B) : m$
- **Injection**:
  - The attacker can send protocol messages: $O(A) \rightarrow B : m$
- **Compromise**:
  - The attacker may directly play one of the protocol roles: $A \rightarrow O : m$

These capabilities are enough to:

- Steal sensitive data (e.g. passwords, bank statements)
- Impersonate principals (e.g. bank clients and bank servers)
- Tamper with sensitive data (e.g. bank transactions)

To prevent such attacks, protocols rely on cryptography
Threat Model: *the network is the attacker*

- **Attacker Principals:** $E$ (eve), $M$ (mallory), $O$ (opponent), \ldots
- Each message is sent over an insecure network: $A \rightarrow B : m$
- **Interception:**
  The attacker can steal protocol messages: $A \rightarrow O(B) : m$
- **Injection:**
  The attacker can send protocol messages: $O(A) \rightarrow B : m$
- **Compromise:**
  The attacker may directly play one of the protocol roles: $A \rightarrow O : m$

- These capabilities are enough to:
  - Steal sensitive data (e.g. passwords, bank statements)
  - Impersonate principals (e.g. bank clients and bank servers)
  - Tamper with sensitive data (e.g. bank transactions)
- To prevent such attacks, protocols rely on cryptography
Cryptographic primitives: symmetric encryption

- **Shared Keys**: $K, K_{AB}, \ldots$
- **Symmetric Encryption**: $\{m\}_n$
- **Symmetric Decryption**: $\text{dec}(m, n)$
  - $\text{dec}(%m, K) = m$
- **Security Property**: Plaintext confidentiality
  - Informally, a ciphertext cannot be decrypted without knowing the key
  - Different formulations in the formal and computational models
- **Examples**: DES, AES, RC4, ...
  - Many concrete details: initialization vectors, block size, streams, \ldots
Cryptographic primitives: asymmetric encryption

- **Private Keys**: sk(A) (for principal A)
- **Public Keys**: pk(A) (for principal A)
  - Public key infrastructure (PKI): Assume that the public keys of all principals are known
- **Asymmetric Encryption**: \( \{m\}_{pk(A)} \)
- **Asymmetric Decryption**: \( \text{dec}(m, sk(A)) \)
  - \( \text{dec}(\{m\}_{pk(A)}, sk(A)) = m \)

- **Security Property**: Plaintext confidentiality
  - Informally, a message encrypted with the public key of A cannot be decrypted without knowing the private key of A
  - Different formulations in the formal and computational models
- **Examples**: RSA, ElGamal, Cramer-Shoup, …
  - Many concrete details: key size, malleability, determinism …
Cryptographic primitives: signature

- **Signature**: \( \text{sig}\{m\}_{sk(A)} \)
- **Verification**: \( \text{verify}(m, n, \text{pk}(A)) \)
  - \( \text{verify}(m, \text{sig}\{m\}_{sk(A)}, \text{pk}(A)) = \text{true} \)

- **Security Property**: Plaintext unforgeability
  - Informally, a signature that can be verified using the public key of \( A \) cannot be created without knowing the private key of \( A \)
  - Different formulations in the formal and computational models

- **Examples**: RSA, DSA, ECDSA, ...
  - Many concrete details: key size, hash function, non-repudiation, ...
Protocol Security Goals

- **Confidentiality**: Can the attacker $O$ learn a secret that is meant to be known only to $A$ and $B$?
Protocol Security Goals

- **Confidentiality**: Can the attacker \( O \) learn a secret that is meant to be known only to \( A \) and \( B \)?
- **Integrity**: Can the attacker \( O \) modify a message from \( A \) and get it accepted by \( B \)?
Protocol Security Goals

- **Confidentiality**: Can the attacker $O$ learn a secret that is meant to be known only to $A$ and $B$?
- **Integrity**: Can the attacker $O$ modify a message from $A$ and get it accepted by $B$?
- **Authentication**: Can the attacker $O$ convince $B$ that it is talking to $A$?
- **Anonymity**: If $A$ wishes to be anonymous during the protocol, can $O$ discover its identity?
- **Non-Repudiation**: If $A$ sends a message to $B$, can it later deny that it sent the message? Can $B$ deny that it received the message?
- **Fairness**: Can one of $A$ or $B$ obtain an unfair advantage before the transaction is completed? Can $A$ obtain a good without paying?
Protocol Security Goals

- **Confidentiality**: Can the attacker $O$ learn a secret that is meant to be known only to $A$ and $B$?
- **Integrity**: Can the attacker $O$ modify a message from $A$ and get it accepted by $B$?
- **Authentication**: Can the attacker $O$ convince $B$ that it is talking to $A$?
- **Anonymity**: If $A$ wishes to be anonymous during the protocol, can $O$ discover its identity?
Protocol Security Goals

- **Confidentiality**: Can the attacker $O$ learn a secret that is meant to be known only to $A$ and $B$?
- **Integrity**: Can the attacker $O$ modify a message from $A$ and get it accepted by $B$?
- **Authentication**: Can the attacker $O$ convince $B$ that it is talking to $A$?
- **Anonymity**: If $A$ wishes to be anonymous during the protocol, can $O$ discover its identity?
- **Non-Repudiation**: If $A$ sends a message to $B$, can it later deny that it sent the message? Can $B$ deny that it received the message?
Protocol Security Goals

- **Confidentiality**: Can the attacker $O$ learn a secret that is meant to be known only to $A$ and $B$?
- **Integrity**: Can the attacker $O$ modify a message from $A$ and get it accepted by $B$?
- **Authentication**: Can the attacker $O$ convince $B$ that it is talking to $A$?
- **Anonymity**: If $A$ wishes to be anonymous during the protocol, can $O$ discover its identity?
- **Non-Repudiation**: If $A$ sends a message to $B$, can it later deny that it sent the message? Can $B$ deny that it received the message?
- **Fairness**: Can one of $A$ or $B$ obtain an unfair advantage before the transaction is completed? Can $A$ obtain a good without paying?
Example Security Goals: E-Voting

- **Eligibility**: only legitimate voters can vote, and only once
- **Vote Privacy**: the fact that a particular voted in a particular way is not revealed to anyone
- **Individual verifiability**: a voter can verify that her vote was really counted
- **Universal verifiability**: the published outcome really is the sum of all the votes
- **Fairness**: no early results can be obtained which could influence the remaining voters
- **Receipt-freeness**: a voter cannot prove that she voted in a certain way (this is important to protect voters from coercion)
- **Coercion-resistance**: same as receipt-freeness, but the coercer interacts with the voter during the protocol, (e.g. by preparing messages)
Example: Towards a secure RPC

Alice (A) wishes to perform an online transaction with her bank (B):

\[ A \rightarrow B : \text{request} \]
\[ B \rightarrow A : \text{response} \]

Security Goals:
- Confidentiality of request and response;
- Integrity of request and response;
- Authentication of \( A \) and \( B \)
Secure RPC: Cryptographic Protocol 1

- Assume that $A$ and $B$ know each other’s public keys $\text{pk}(A)$, $\text{pk}(B)$

$$
A \rightarrow B : \{\text{request}\}_{\text{pk}(B)} \\
B \rightarrow A : \{\text{response}\}_{\text{pk}(A)}
$$

- Does this protocol provide confidentiality for request and response?
Assume that $A$ and $B$ know each other’s public keys $\text{pk}(A)$, $\text{pk}(B)$

$A \to B : \{\text{request}\}_{\text{pk}(B)}$

$B \to A : \{\text{response}\}_{\text{pk}(A)}$

Does this protocol provide confidentiality for request and response?

Does it authenticate $A$ and $B$ to each other?
Secure RPC: Cryptographic Protocol 1

- Assume that $A$ and $B$ know each other’s public keys $pk(A)$, $pk(B)$

  $A \rightarrow B : \{\text{request}\}_{pk(B)}$
  $B \rightarrow A : \{\text{response}\}_{pk(A)}$

- Does this protocol provide confidentiality for request and response?
- Does it authenticate $A$ and $B$ to each other?
- No, the attacker can send arbitrary requests and responses

  $O(A) \rightarrow B : \{\text{request}\}_{pk(B)}$

- Does adding $A$ and $B$ to the messages help?

  $A \rightarrow B : \{\langle A, \text{request}\rangle\}_{pk(B)}$
  $B \rightarrow A : \{\langle B, \text{response}\rangle\}_{pk(A)}$
Protocol 2: Adding Signatures

- Sign the ciphertexts with the sender’s private key

  \[ A \rightarrow B : \{ \text{request} \}_{\text{pk}(B)}, \text{sig}\{\{ \text{request} \}_{\text{pk}(B)} \}_{\text{sk}(A)} \]

  \[ B \rightarrow A : \{ \text{response} \}_{\text{pk}(A)}, \text{sig}\{\{ \text{response} \}_{\text{pk}(A)} \}_{\text{sk}(B)} \]

- Does this ensure message integrity? sender authentication?
- What about replay attacks?

  \[ O(A) \rightarrow B : \{ \text{request} \}_{\text{pk}(B)}, \text{sig}\{\{ \text{request} \}_{\text{pk}(B)} \}_{\text{sk}(A)} \]

  \[ O(A) \rightarrow B : \{ \text{request} \}_{\text{pk}(B)}, \text{sig}\{\{ \text{request} \}_{\text{pk}(B)} \}_{\text{sk}(A)} \]

  \[ O(A) \rightarrow B : \{ \text{request} \}_{\text{pk}(B)}, \text{sig}\{\{ \text{request} \}_{\text{pk}(B)} \}_{\text{sk}(A)} \]

- What if request = “Transfer 1000EUR to O”?
Protocol 3: Preventing replays with timestamps

- To prevent replays we could add a timestamp to each message:
  \[ A \rightarrow B : \{ \langle T, \text{request} \rangle \}_{pk(B)}, \text{sig}\{ \{ \langle T, \text{request} \rangle \}_{pk(B)} \}_{sk(A)} \]

- \( B \) should reject messages that are older than some threshold \( \Delta \)
- This requires the clocks at \( A \) and \( B \) to be synchronized
- It still leaves a window of opportunity for replay attacks (\( \Delta \))
Protocol 4: Preventing replays with nonces

- An alternative is to include a challenge-response mechanism based on a fresh randomly-generated nonce.
- $B$ generates a nonce and sends it to $A$ which includes it in the request:

  $$
  B \rightarrow A : \{N\}_{pk(A)} \\
  A \rightarrow B : \{\langle N, \text{request} \rangle\}_{pk(B)}, \text{sig}\{\{\langle N, \text{request} \rangle\}_{pk(B)} \}^{sk(A)} \\
  B \rightarrow A : \{\langle N, \text{response} \rangle\}_{pk(A)}, \text{sig}\{\{\langle N, \text{response} \rangle\}_{pk(A)} \}^{sk(B)}
  $$

- $B$ rejects any response that does not include the correct nonce.
- Here, the nonce acts as a unique session identifier.
- More generally, each principal may use its own nonce ($N_A, N_B$); we will see an example in the Needham Schroeder protocol later.
Protocol 2: Response confidentiality

- Suppose request contains the login/password for A’s account
  - We assume that O does not know the password
- Suppose response is A’s current bank statement
- Does protocol 2 keep response confidential?

\begin{align*}
A &\rightarrow B : \{\text{request}\}_{\text{pk}(B)} , \text{sig}\{\{\text{request}\}_{\text{pk}(B)}\}_{\text{sk}(A)} \\
B &\rightarrow A : \{\text{response}\}_{\text{pk}(A)} , \text{sig}\{\{\text{response}\}_{\text{pk}(A)}\}_{\text{sk}(B)}
\end{align*}
Protocol 2: Response confidentiality

- Suppose request contains the login/password for A’s account
  - We assume that O does not know the password
- Suppose response is A’s current bank statement
- Does protocol 2 keep response confidential?

\[
\begin{align*}
A & \rightarrow B : \text{request}^{pk(B)}, \text{sig}\{\text{request}^{pk(B)}\}^{sk(A)} \\
B & \rightarrow A : \text{response}^{pk(A)}, \text{sig}\{\text{response}^{pk(A)}\}^{sk(B)}
\end{align*}
\]

- No! We show a *man-in-the-middle attack*
Protocol 2: Man-in-the-middle Attack

- Does protocol 2 keep response confidential?
  
  \[
  A \rightarrow B : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)}
  
  B \rightarrow A : \{\text{response}\}_{pk(A)}, \text{sig}\{\{\text{response}\}_{pk(A)}\}_{sk(B)}
  \]

- Suppose \(O\) intercepts \(A\)'s request to \(B\)
  
  \[
  A \rightarrow O(B) : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)}
  \]

- Hence \(O\) obtains the secret response (bank statement of \(A\))

- Exercise: Show that adding nonces as in protocol 4 does not help
Protocol 2: Man-in-the-middle Attack

- Does protocol 2 keep response confidential?
  
  $A \rightarrow B : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)}$
  
  $B \rightarrow A : \{\text{response}\}_{pk(A)}, \text{sig}\{\{\text{response}\}_{pk(A)}\}_{sk(B)}$

- Suppose $O$ intercepts $A$’s request to $B$
  
  $A \rightarrow O(B) : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)}$

- Then $O$ replaces the signature with its own and forwards it
  
  $O \rightarrow B : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(O)}$

- Hence $O$ obtains the secret response (bank statement of $A$)

- Exercise: Show that adding nonces as in protocol 4 does not help
Protocol 2: Man-in-the-middle Attack

- Does protocol 2 keep response confidential?
  
  $A \rightarrow B : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)}$
  
  $B \rightarrow A : \{\text{response}\}_{pk(A)}, \text{sig}\{\{\text{response}\}_{pk(A)}\}_{sk(B)}$

- Suppose $O$ intercepts $A$’s request to $B$
  
  $A \rightarrow O(B) : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)}$

- Then $O$ replaces the signature with its own and forwards it
  
  $O \rightarrow B : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(O)}$

- $B$ thinks that this request came from $O$ and responds with:
  
  $B \rightarrow O : \{\text{response}\}_{pk(O)}, \text{sig}\{\{\text{response}\}_{pk(O)}\}_{sk(B)}$

- Hence $O$ obtains the secret response (bank statement of $A$)

- Exercise: Show that adding nonces as in protocol 4 does not help
Protocol 2: Guessing Attacks

- The attacks so far are *logical* or *symbolic* attacks.
- We now consider a *computational* attack on request confidentiality.
- What can $O$ learn about request?

\[
A \rightarrow B : \{\text{request}\}_{\text{pk}(B)}, \text{sig}\{\{\text{request}\}_{\text{pk}(B)}\}_{\text{sk}(A)}
\]

Suppose request is a password of 8 characters? Can $O$ guess it?

**Password Guessing:** $O$ guesses request and sends it to $B$; if $B$ responds then the password must be correct.

**Limited practicality:** 3 guesses and you’re out.

**Offline/Passive Attack:** Suppose the encryption algorithm is deterministic.

**Brute-force Search:** $O$ generates all strings of 8 characters, encrypts them using $\text{pk}(B)$, and checks whether the result matches the message.

**Dictionary Attack:** $O$ starts from a dictionary of commonly used passwords (e.g. English phrases).
Protocol 2: Guessing Attacks

- The attacks so far are *logical* or *symbolic* attacks
- We now consider a *computational* attack on request confidentiality
- What can $O$ learn about request?

\[ A \rightarrow B : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)} \]

- Suppose request is a password of 8 characters? Can $O$ guess it?
  - *Password Guessing*: $O$ guesses request and sends it to $B$; if $B$ responds then the password must be correct.
  - Limited practicality: 3 guesses and you’re out
Protocol 2: Guessing Attacks

- The attacks so far are *logical* or *symbolic* attacks
- We now consider a *computational* attack on request confidentiality
- What can $O$ learn about request?

$$A \rightarrow B : \{\text{request}\}_{pk(B)}, \text{sig}\{\{\text{request}\}_{pk(B)}\}_{sk(A)}$$

- Suppose request is a password of 8 characters? Can $O$ guess it?
  - *Password Guessing*: $O$ guesses request and sends it to $B$; if $B$ responds then the password must be correct.
  - Limited practicality: 3 guesses and you’re out

- Offline/Passive Attack:
  - Suppose the encryption algorithm is deterministic
  - *Brute-force Search*: $O$ generates all strings of 8 characters, encrypts them using $pk(B)$, and checks whether the result matches the message
  - *Dictionary Attack*: $O$ starts from a dictionary of commonly used passwords (e.g. English phrases).
From attacks to proofs

- Can we be confident that there are no more attacks on the protocol?
- How can we mathematically prove that the protocol satisfies its security goals?
  - What does $A \rightarrow B : m$ mean? It specifies the message but now what $A$ and $B$ must do
  - How do we specify security goals?
  - How do we encode the threat model?
- Our informal notation is adequate for finding and explaining attacks
- To precisely state and prove security theorems about cryptographic protocols, we need to move to a more formal setting.
The formal model or “Dolev-Yao model” is due to Needham and Schroeder [1978] and Dolev and Yao [1983].

- The cryptographic primitives are blackboxes.
- The messages are terms on these primitives.
  - $\{m\}_k$ encryption of the message $m$ with key $k$,
  - $(m_1, m_2)$ pairing of messages $m_1$ and $m_2$, ...
- The attacker is restricted to compute only using these primitives.
  ⇒ perfect cryptography assumption

One can add equations between primitives, but in any case, one makes the hypothesis that the only equalities are those given by these equations.

This model makes automatic proofs relatively easy (AVISPA, ProVerif, ...).
The computational model has been developed at the beginning of the 1980’s by Goldwasser, Micali, Rivest, Yao, and others.

- The messages are bitstrings.
- The cryptographic primitives are functions on bitstrings.
- The attacker is any probabilistic (polynomial-time) Turing machine.

This model is much more realistic than the formal model, but until recently proofs were only manual.
The computational model is still just a model, which does not exactly match reality.

In particular, it ignores side channels:

- timing
- power consumption
- noise
- physical attacks against smart cards

which can give additional information.

In this course, we will mostly ignore side channels.
Compute the set of all terms that the attacker can obtain.

This set is infinite:
- The attacker can generate messages of unbounded size.
- The number of sessions of the protocol is unbounded.
Complexity

- Bounded messages and number of sessions
  - $\Rightarrow$ finite state
  - Model checking: FDR [Lowe, TACAS’96]
- Bounded number of sessions but unbounded messages
  - $\Rightarrow$ insecurity is typically NP-complete
  - Constraint solving: Cl-AtSe, integrated in AVISPA
    Extensions of model checking: OFMC, integrated in AVISPA
- Unbounded messages and number of sessions
  - $\Rightarrow$ the problem is undecidable
Solutions to undecidability

- Rely on user interaction
  - Interactive theorem proving, Isabelle [Paulson, JCS’98]
- Use approximations
  - Abstract interpretation [Monniaux, SCP’03], TA4SP integrated in AVISPA
  - Typing [Abadi, JACM’99], [Gordon, Jeffrey, CSFW’02] (Sometimes also relies on type annotations by the user.)
- Allow non-termination

ProVerif uses approximations and allows non-termination.
Numerous attacks have already been obtained.

An attack in the formal model immediately implies an in the computational model (and a practical attack).

- A proof in the formal model does not always imply a proof in the computational model (see next).

Allows us to perform automatic verification.
Proofs in the computational model

- Manual proofs by cryptographers:
  - proofs by sequences of games [Shoup, Bellare&Rogaway]

- Automation:
  - CryptoVerif
  - CertiCrypt, framework within Coq
  - Typing
Link between the two models

- **Computational soundness** theorems:
  
  Proof in the formal model $\Rightarrow$ proof in the computational model

  modulo additional assumptions.

  Approach pioneered by Abadi & Rogaway [2000]; many works since then.
Indirect approach to automating computational proofs:

1. Automatic formal protocol verifier

2. Computational proof in the formal model $\rightarrow$ computational soundness $\rightarrow$ proof in the computational model
- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names**: $a$, $b$, $c$, … (used for keys, nonces, channels)
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names**: $a, b, c, \ldots$ (used for keys, nonces, channels)
- **Messages**: $M, N, \ldots$
  - **Constructors**: $\langle m, n \rangle$ (pairing), $\{m\}_k$, $\text{sig}\{m\}_k$, $\text{pk}(m)$
  - **Destructors**: $\text{proj}_1(m)$, $\text{proj}_2(m)$, $\text{dec}(m, k)$, $\text{verify}(m, s, k)$
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names**: $a, b, c, \ldots$ (used for keys, nonces, channels)
- **Messages**: $M, N, \ldots$
  - **Constructors**: $\langle m, n \rangle$ (pairing), $\{ m \}_k$, $\text{sig}\{ m \}_k$, $\text{pk}(m)$
  - **Destructors**: $\text{proj}_1(m)$, $\text{proj}_2(m)$, $\text{dec}(m, k)$, $\text{verify}(m, s, k)$
- **Processes**: $P, Q, R, \ldots$
  - $P, Q, R ::= \quad \text{Processes}$
  - $0 \quad \text{null process}$
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names**: $a, b, c, \ldots$ (used for keys, nonces, channels)
- **Messages**: $M, N, \ldots$
  - **Constructors**: $\langle m, n \rangle$ (pairing), $\{m\}_k$, $\text{sig}\{m\}_k$, $\text{pk}(m)$
  - **Destructors**: $\text{proj}_1(m)$, $\text{proj}_2(m)$, $\text{dec}(m, k)$, $\text{verify}(m, s, k)$
- **Processes**: $P, Q, R, \ldots$
  
  $P, Q, R ::=$
  
  0
  
  new $a. P$
  
  Processes
  null process
  fresh name generation
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names**: $a, b, c, \ldots$ (used for keys, nonces, channels)
- **Messages**: $M, N, \ldots$
  - **Constructors**: $\langle m, n \rangle$ (pairing), $\{ m \}_k$, $\text{sig}\{ m \}_k$, $\text{pk}(m)$
  - **Destructors**: $\text{proj}_1(m)$, $\text{proj}_2(m)$, $\text{dec}(m, k)$, $\text{verify}(m, s, k)$
- **Processes**: $P, Q, R, \ldots$
  
  $P, Q, R ::=$
  
  0
  
  new $a.P$
  
  $\text{in}(c, x).P$
  
  Processes
  
  null process
  
  fresh name generation
  
  message input (continue as P)
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names:** $a$, $b$, $c$, ... (used for keys, nonces, channels)
- **Messages:** $M$, $N$, ...
  - **Constructors:** $\langle m, n \rangle$ (pairing), $\{ m \}_k$, $\text{sig}\{ m \}_k$, $\text{pk}(m)$
  - **Destructors:** $\text{proj}_1(m)$, $\text{proj}_2(m)$, $\text{dec}(m, k)$, $\text{verify}(m, s, k)$
- **Processes:** $P$, $Q$, $R$, ...

\[
\begin{align*}
P, Q, R &::= \\
0 &\quad \text{null process} \\
\text{new } a.P &\quad \text{fresh name generation} \\
in(c, x).P &\quad \text{message input (continue as P)} \\
\text{out}(c, M).P &\quad \text{message output (continue as P)}
\end{align*}
\]
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names**: \( a, b, c, \ldots \) (used for keys, nonces, channels)
- **Messages**: \( M, N, \ldots \)
  - **Constructors**: \( \langle m, n \rangle \) (pairing), \( \{ m \}_k \), \( \text{sig}\{ m \}_k \), \( \text{pk}(m) \)
  - **Destructors**: \( \text{proj}_1(m) \), \( \text{proj}_2(m) \), \( \text{dec}(m, k) \), \( \text{verify}(m, s, k) \)
- **Processes**: \( P, Q, R, \ldots \)
  \[
P, Q, R ::= \\
 0 \\
\text{new } a.P \\
\text{in}(c, x).P \\
\text{out}(c, M).P \\
\text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q
\]
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- Names: $a, b, c, \ldots$ (used for keys, nonces, channels)
- Messages: $M, N, \ldots$
  - Constructors: $\langle m, n \rangle$ (pairing), $\{m\}_k$, $\text{sig}\{m\}_k$, $\text{pk}(m)$
  - Destructors: $\text{proj}_1(m)$, $\text{proj}_2(m)$, $\text{dec}(m, k)$, $\text{verify}(m, s, k)$
- Processes: $P, Q, R, \ldots$
  \[
P, Q, R ::=\]
  \[
  0 \quad \text{null process} \\
  \text{new } a.P \quad \text{fresh name generation} \\
  \text{in}(c, x).P \quad \text{message input (continue as } P) \\
  \text{out}(c, M).P \quad \text{message output (continue as } P) \\
  \text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q \quad \text{destructor application} \\
  \text{if } M = N \text{ then } P \text{ else } Q \quad \text{conditional}
\]
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus \[\text{[Abadi, Fournet, 2000]}\]
- **Names**: \(a, b, c, \ldots\) (used for keys, nonces, channels)
- **Messages**: \(M, N, \ldots\)
  - **Constructors**: \(\langle m, n \rangle\) (pairing), \(\{m\}_k\), \(\text{sig}\{m\}_k\), \(\text{pk}(m)\)
  - **Destructors**: \(\text{proj}_1(m)\), \(\text{proj}_2(m)\), \(\text{dec}(m, k)\), \(\text{verify}(m, s, k)\)
- **Processes**: \(P, Q, R, \ldots\)
  - \(P, Q, R ::=\)
    - \(0\)
    - \(\text{new } a.P\)
    - \(\text{in}(c, x).P\)
    - \(\text{out}(c, M).P\)
    - \(\text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q\)
    - \(\text{if } M = N \text{ then } P \text{ else } Q\)
    - \(P|Q\)
Formal protocol modeling: A small process calculus

- Simplified version of the applied pi calculus [Abadi, Fournet, 2000]
- **Names**: $a, b, c, \ldots$ (used for keys, nonces, channels)
- **Messages**: $M, N, \ldots$
  - **Constructors**: $\langle m, n \rangle$ (pairing), $\{m\}_k$, sig$\{m\}_k$, pk($m$)
  - **Destructors**: proj$_1(m)$, proj$_2(m)$, dec($m, k$), verify($m, s, k$)
- **Processes**: $P, Q, R, \ldots$
  - $P, Q, R ::=$
    - 0
    - new $a.P$
    - in($c, x$).$P$
    - out($c, M$).$P$
    - let $x = g(M_1, \ldots, M_n)$ in $P$ else $Q$
    - if $M = N$ then $P$ else $Q$
    - $P|Q$
    - $!P$
Example

- Assume that A and B know each other’s public keys \( pk(A), pk(B) \)

\[
A \rightarrow B : \{\text{request}\}_{pk(B)}
\]

\[
B \rightarrow A : \{\text{response}\}_{pk(A)}
\]

- Write processes for A and B
Needham-Schroeder (public-key) Protocol
Needham-Schroeder’s Protocol (1978)

- $A \rightarrow B : \{A, N_a\}_{pk(B)}$
- $B \rightarrow A : \{N_a, N_b\}_{pk(A)}$
- $A \rightarrow B : \{N_b\}_{pk(B)}$

Questions:
- Is $N_b$ secret between $A$ and $B$?
- When $B$ receives $\{N_b\}_{pk(B)}$, does this message really come from $A$?

Attack:
An attack was found 17 years after its publication! [Lowe 96]
Needham-Schroeder’s Protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \{A, N_a\}_{pk(B)} \\
B & \rightarrow A : \{N_a, N_b\}_{pk(A)} \\
A & \rightarrow B : \{N_b\}_{pk(B)}
\end{align*}
\]

Questions

Is \(N_b\) secret between \(A\) and \(B\)?

When \(B\) receives \(\{N_b\}_{pk(B)}\), does this message really come from \(A\)?

Attack

An attack was found 17 years after its publication! [Lowe 96]
Needham-Schroeder’s Protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \quad \{A, N_a\}_{pk(B)} \\
B & \rightarrow A : \quad \{N_a, N_b\}_{pk(A)} \\
A & \rightarrow B : \quad \{N_b\}_{pk(B)}
\end{align*}
\]

Questions
- Is \(N_b\) secret between \(A\) and \(B\)?
- When \(B\) receives \(\{N_b\}_{pk(B)}\), does this message really come from \(A\)?

Attack
An attack was found 17 years after its publication! [Lowe 96]
Needham-Schroeder’s Protocol (1978)

\[
\begin{align*}
A &\to B: \quad \{A, N_a\}_{pk(B)} \\
B &\to A: \quad \{N_a, N_b\}_{pk(A)} \\
A &\to B: \quad \{N_b\}_{pk(B)}
\end{align*}
\]

Questions

Is \(N_b\) secret between \(A\) and \(B\)?

When \(B\) receives \(\{N_b\}_{pk(B)}\), does this message really come from \(A\)?

Attack

An attack was found 17 years after its publication! [Lowe 96]

Karthikeyan Bhargavan (INRIA)

Introduction to cryptographic protocols

September 2013 39 / 43
Needham-Schroeder’s Protocol (1978)

\[ A \rightarrow B : \{A, Na\}_{pk(B)} \]
\[ B \rightarrow A : \{Na, Nb\}_{pk(A)} \]
\[ A \rightarrow B : \{Nb\}_{pk(B)} \]

Questions

- Is \(Nb\) secret between \(A\) and \(B\)?
- When \(B\) receives \(\{Nb\}_{pk(B)}\), does this message really comes from \(A\)?
Needham-Schroeder’s Protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \{A, N_a\}_{pk(B)} \\
B & \rightarrow A : \{N_a, N_b\}_{pk(A)} \\
A & \rightarrow B : \{N_b\}_{pk(B)}
\end{align*}
\]

Questions

- Is \(N_b\) secret between \(A\) and \(B\) ?
- When \(B\) receives \(\{N_b\}_{pk(B)}\), does this message really comes from \(A\) ?

Attack

An attack was found 17 years after its publication! [Lowe 96]
Example: Man in the middle attack

- Involving 2 sessions in parallel,
- An honest agent has to initiate a session with \( I \).

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent A</td>
<td>Intruder I</td>
<td>Agent B</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
A & \rightarrow B & : & \{A, N_a\}_{pk(B)} \\
B & \rightarrow A & : & \{N_a, N_b\}_{pk(A)} \\
A & \rightarrow B & : & \{N_b\}_{pk(B)}
\end{align*}
\]
Example: Man in the middle attack

\[
\{A, N_a\}_{pk(I)} \quad \{A, N_a\}_{pk(B)}
\]

Agent A

Intruder I

Agent B

A → B : \{A, N_a\}_{pk(B)}
B → A : \{N_a, N_b\}_{pk(A)}
A → B : \{N_b\}_{pk(B)}
Example: Man in the middle attack

\[ \{A, N_a\}_{pk(I)} \]
\[ \{N_a, N_b\}_{pk(A)} \]

Agent A

Intruder I

\[ \{A, N_a\}_{pk(B)} \]
\[ \{N_a, N_b\}_{pk(A)} \]

Agent B

\[ A \rightarrow B : \{A, N_a\}_{pk(B)} \]
\[ B \rightarrow A : \{N_a, N_b\}_{pk(A)} \]
\[ A \rightarrow B : \{N_b\}_{pk(B)} \]
Example: Man in the middle attack

Agent A

\{A, Na\}_{pk(I)}

\{Na, Nb\}_{pk(A)}

\{Nb\}_{pk(I)}

Intruder I

Agent B

\{A, Na\}_{pk(B)}

\{Na, Nb\}_{pk(A)}

\{Nb\}_{pk(B)}

A → B : \{A, Na\}_{pk(B)}

B → A : \{Na, Nb\}_{pk(A)}

A → B : \{Nb\}_{pk(B)}
Example: Man in the middle attack

Agent A

\[ \{ A, N_a \} \text{pk}(I) \]
\[ \{ N_a, N_b \} \text{pk}(A) \]
\[ \{ N_b \} \text{pk}(I) \]

Intruder I

\[ \{ A, N_a \} \text{pk}(B) \]
\[ \{ N_a, N_b \} \text{pk}(A) \]
\[ \{ N_b \} \text{pk}(B) \]

Agent B

\[ \{ N_a, N_b \} \text{pk}(A) \]
\[ \{ N_b \} \text{pk}(B) \]

The intruder knows \( N_b \),

When B finishes his session (apparently with A), A has never talked with B.

A \rightarrow B : \{ A, N_a \} \text{pk}(B)
B \rightarrow A : \{ N_a, N_b \} \text{pk}(A)
A \rightarrow B : \{ N_b \} \text{pk}(B)
A → B : \{A, N_a\}_{pk(B)}
B → A : \{N_a, N_b\}_{pk(A)}
A → B : \{N_b\}_{pk(B)}

Example (Exercise)
Propose a fix for the Needham-Schroeder protocol.
Exercise

\[
\begin{align*}
A & \rightarrow B : \{ A, N_a \}_{pk(B)} \\
B & \rightarrow A : \{ N_a, N_b \}_{pk(A)} \\
A & \rightarrow B : \{ N_b \}_{pk(B)}
\end{align*}
\]

Example (Exercise)

Write processes for \( A \) and \( B \) in the process calculus.

Example (Exercise)

Write the attacker process and show how the attack works.

Example (Exercise)

Write the secure RPC protocol in the process calculus and demonstrate its attacks.
Exercise

- Who are the principals?
- What is their initial state? What do they share?
- What is the sequence of actions of each principal?
- What are the security goals?