CRYPTOGRAPHIC PROTOCOLS 2016, LECTURE 13

SIGMA PROTOCOLS FOR DL

HELGER LIPMAA, UNIVERSITY OF TARTU
UP TO NOW

- Introduction to the field
- Secure computation protocols
- Introduction to malicious model
- $\Sigma$-protocols: basics
Σ-PROTOCOLS: UP TO NOW

- We started from a very simple protocol for GI
  - "intuitively" secure: complete, sound, ZK
- Formalized security requirements
  - based on this concrete example
- Big promise: this is useful in general
THIS TIME

- Σ-protocols: short reminder
- Σ-protocols for DL based languages
- Composition of Σ-protocols
- Σ-protocols for everything interesting
**REMINDER: Σ-PROTOCOLS**

1st message: commitment \( a \)

2nd message: challenge \( c \)

3rd message: response \( z \)

\( x, \omega \)

**Requirement:** \( c \) is chosen from publicly known challenge set \( C \) randomly. (Does **not** depend on \( a \! \))

**Terminology:** public coin protocol
REMINDER: Σ-PROTOCOLS

1. Completeness
2. Special Soundness
3. Special Honest-Verifier ZK (SHVZK)

1st message: commitment $a$
2nd message: challenge $c$
3rd message: response $z$

Accepts iff prover knows $\omega$ such that $(x, \omega) \in R$
REMINDER: AUTHENTICATION

pk, sk

1st message: commitment $a$

2nd message: challenge $c$

3rd message: response $z$

pk

Accepts iff prover knows $sk$ such that $(pk, sk) \in Gen$

GI protocol: $sk = \omega = \text{secret isomorphism}$. Verifier convinced prover knows the secret key without any side information being leaked
PROBLEM WITH GI

- **Inefficient** (for say authentication):
  - having graphs as PK/SK is not efficient
  - adjacency matrix: $O(n^2)$ bits

- **Knowledge error** $\kappa = 1/2$

- Need to parallel-repeat protocol 40+ times to amplify

Also we have many other elements of the protocols based on Elgamal. Better stick to the same cryptosystem
NOTATION

- $P$ has input $(x, \omega)$, $V$ has input $x$

- $\Sigma$-protocol $(P, V)$ is a **proof of knowledge** that $P$ knows $\omega$ such that $(x, \omega) \in R$ for a public language $L$/relation $R$

  - $x \in L \iff \exists \omega \ (x, \omega) \in R$ // $\omega$ is short

- Common notation: $\text{PK} \ (\omega: R \ (x, \omega))$

  - Secret values denoted by Greek letters

- We will now see (for Elgamal): $\text{PK} \ (\rho: b = g^\rho)$
POK: KNOWLEDGE OF DL

$h = g^q$, $q$

1st message: commitment $a$

2nd message: challenge $c$

3rd message: response $z$

Accepts iff prover knows $q$ such that $h = g^q$

**Authentication:** Proof of Knowledge of secret key corresponding to Elgamal PK $h$.

**General interpretation:** PK of a discrete logarithm
IDEA

- GI: knowing isomorphism $G_1 \rightarrow G_2 \iff$ (knowing isomorphism $G_1 \rightarrow H \iff$ knowing isomorphism $G_2 \rightarrow H$ for random $H$)
  - Prover knows 0, 1 or 3 of isomorphisms

- DL: knowing DL of $y = g^o \iff$ (knowing DL of $g^r \iff$ knowing DL of $g^{o+r}$ for random $r$)
  - Prover knows 0, 1 or 3 of DL-s
  - It is a bit like secret sharing

\[
\begin{align*}
\phi &: (G_1, G_2) \\
\psi &: (G_1, H) \\
\psi \phi^{-1} &: (G_2, H) \\
\rho &: h = g^o \\
r &: a = g^r \\
r + q &: ah = g^{r+q}
\end{align*}
\]
QUIZ: KNOWLEDGE OF DL

\[ h = g^q, \quad q \]

\[ r \leftarrow \mathbb{Z}_p \]
\[ a \leftarrow g^r \]
\[ c \leftarrow \{0, 1\} \]

Accept if \( g^z = ah^c \)

\[ z \leftarrow r + cq \]

\[ z = r + cq \] plays the role of \( G_c \)

Completeness: \( g^z = g^r + cq = ah^c \)
**QUIZ: KNOWLEDGE OF DL**

\[ h = g^q, Q \]

\[ r \leftarrow \mathbb{Z}_p \]
\[ a \leftarrow g^r \]
\[ a \]
\[ c \leftarrow \{0, 1\} \]
\[ b \]
\[ z \leftarrow r + cQ \]

**Special soundness.**

**Extractor** \( K(a, c, z, c^* \neq c, z^*) \):

1. Return \( q \leftarrow (z - z^*) / (c - c^*) \)

**Analysis.** Divide the first verification equation with the second one:

\[ g^z - z^* = b^{c - c^*} \]

Since \( c \neq c^* \), \( b = g^{(z - z^*) / (c - c^*)} \), thus extractor retrieves \( q \) correctly.
**QUIZ: KNOWLEDGE OF DL**

**Analysis.** First, \((a, c, z)\) is accepting since \(a\) was generated so that the verification equation would accept. (tautology.)

Second. In real execution \((a, c, z) \in G \times \{0, 1\} \times \mathbb{Z}_p\) is completely random except that the verification equation holds. Thus in real execution one can start by generating \(c \leftarrow \{0, 1\}, z \leftarrow \mathbb{Z}_p\), and then setting \(a \leftarrow g^z / h^c\). Thus simulator's output is exactly from the same distribution as the real protocol's output.

**SHVZK. Simulator** \(S(c)\):

1. Generate \(z \leftarrow \mathbb{Z}_p\)
2. Set \(a \leftarrow g^z / h^c\)
3. Return \((a, z)\)
ON KNOWLEDGE ERROR

- We "hid" most of the technicalities
  - "expected" poly-time extractor, etc
- **Important**: \( \kappa = \frac{1}{2} \)
  - \( P^* \) can guess \( V \)'s challenge \( c \) with probability \( \frac{1}{2} \)
  - We can decrease \( \kappa \) by using security amplification from the last lecture
- However, there is a much better way
Quiz: Improving \( K \)

- **Question**: how would you improve \( K \)?
  - without security amplification!

- **Hint 1**:
  - \( K = \frac{1}{|C|} \)

- **Hint 2**: the previous protocol did never require to have \(|C| = 2\)
QUIZ: KNOWLEDGE OF DL

$h = g^o, o$

$r \leftarrow \mathbb{Z}_p$

$a \leftarrow g^r$

$c \leftarrow \{0, 1\}^s$

$z \leftarrow r + cQ$

Accept if $g^z = ah^c$

Completeness: $g^z = g^r + cQ = ah^o$
SECURITY PROOF

- All parts of security proof are same as before

- **Only exception:**
  - In analysis of SHVZK, we get \( c \leftarrow \{0, 1\}^s \) both in the "protocol" and "simulator" case
  - Simulator construction does not change

- **Knowledge error:** \( \kappa = 2^{-s} \) // probability \( P^* \) guesses \( c \)
REMARKS

- In practice $s = 40$ may suffice
  - Information-theoretic limit, not computational
- However, choosing $s = 80$ does not make the protocol significantly less efficient
- This knowledge-of-DL (with variable $s$) $\Sigma$-protocol was proposed by Schnorr in 1991
Another reason $\Sigma$-protocols are useful:

- they are extremely easy to compose
- ... in a black-box way

**Automatic rules:**

1. $\text{PK} \left( R_1(x, \omega_1) \right), \text{PK} \left( R_2(x, \omega_2) \right) \rightarrow \text{PK} \left( (\omega_1, \omega_2): R_1(x, \omega_1) \land R_2(x, \omega_2) \right)$

2. $\text{PK} \left( R_1(x, \omega_1) \right), \text{PK} \left( R_2(x, \omega_2) \right) \rightarrow \text{PK} \left( \omega: R_1(x, \omega) \lor R_2(x, \omega) \right)$

**Given those two rules (and suitable starting protocols), can "automatically" construct $\Sigma$-protocols for all languages in $\text{NP}$**
"AND" COMPOSITION

- Assume you know both $\omega_1$ and $\omega_2$ s.t. $R_i(x, \omega_i)$
- Run $PK(\omega_1: R_1(x, \omega_1))$ and $PK(\omega_2: R_2(x, \omega_2))$ in parallel
  - Possible, since we know both witnesses
- Optimization:
  - let $V$ to use the same $c$ in both cases
  - assuming they come from the same set

this optimization is actually needed for some functionality
"AND" COMPOSITION

\[(x, \omega_i): \text{s.t. } R_i(x, \omega_i)\]

\[a_1 \leftarrow P_1(x; r_1)\]
\[a_2 \leftarrow P_2(x; r_2)\]
\[(a_1, a_2)\]

\[z_1 \leftarrow P_1(x, \omega_1, c; r_1)\]
\[z_2 \leftarrow P_2(x, \omega_2, c; r_2)\]
\[(z_1, z_2)\]

\[c \leftarrow C\]

Accept if both \(V_1(x, a_1, c_1, z_1)\) and \(V_2(x, a_2, c_2, z_2)\) accept
"AND": COMPLETENESS

Completeness: Since both $V_1$ and $V_2$ accept, also $V$ accepts
\[ (x, \omega_i): \text{s.t. } R_i(x, \omega_i) \]

\[ a_1 \leftarrow P_1(x; r_1) \]
\[ a_2 \leftarrow P_2(x; r_2) \]
\[ (a_1, a_2) \]

\[ a_1 \leftarrow P_1(x; r_1) \]
\[ a_2 \leftarrow P_2(x; r_2) \]
\[ (a_1, a_2) \]

\[ c \leftarrow C \]
\[ z_1 \leftarrow P_1(x, \omega_1, c; r_1) \]
\[ z_2 \leftarrow P_2(x, \omega_2, c; r_2) \]
\[ (z_1, z_2) \]

Accept if both \( V_1(x, a_1, c_1, z_1) \) and \( V_2(x, a_2, c_2, z_2) \) accept

**Special soundness.**

**Extractor** \( K(a_1, a_2, c, z_1, z_2, c^*, z_1^*, z_2^*) \):

1. \( \omega_1 \leftarrow K_1(a_1, c, z_1, c^*, z_1^*) \)
2. \( \omega_2 \leftarrow K_2(a_2, c, z_2, c^*, z_2^*) \)
3. Return \((\omega_1, \omega_2)\)
REMKS

- We will see two different types of AND compositions

1. \( \omega_1 \) and \( \omega_2 \) are independent random variables
   - **Example:** \( \text{PK} ((\alpha, \beta): e_1 = h^\alpha \land e_2 = g^\beta) \)

2. Both protocols use the same secret
   - **Example:** \( \text{PK} (\alpha: e_1 = h^\alpha \land e_2 = g^\alpha) \)

- Not a completely automatic composition: have to prove special soundness every single time
POK: DDH WITNESS

\[ L = \{(g, h, e_1, e_2), \text{s.t.} \ (e_1, e_2) = (h^c, g^c) \text{ for some } c\} \]

\[ (g, h, e_1, e_2), \ c \]

\[ (a_1, a_2) \leftarrow (h^r, g^r) \]

\[ z \leftarrow r + cQ \]

\[ (a_1, a_2) \]

\[ c \leftarrow C \]

Second type AND composition of two DL POKs

Accept if \((h^z, g^z) = (a_1e_1^c, a_2e_2^c)\)
DDH WITNESS: COMPLETENESS

$L = \{(g, h, e_1, e_2), \text{s.t.} (e_1, e_2) = (h^\rho, g^\rho) \text{ for some } \rho\}$

Completeness. Clear, since $(a_1, c, z)$ is accepting view for PK $(\rho: e_1 = h^\rho)$ and $(a_2, c, z)$ is accepting view for PK $(\rho: e_2 = g^\rho)$

Second type AND composition of two DL POKs

Accept if $(h^z, g^z) = (a_1e_1^c, a_2e_2^c)$
DDH WITNESS: SPEC. SOUNDNESS

L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (h^\rho, g^\rho) \text{ for some } \rho\}\n
(g, h, e_1, e_2), \rho

r \leftarrow \mathbb{Z}_p

(a_1, a_2) \leftarrow (h^r, g^r)

(a_1, a_2)

c \leftarrow C

z \leftarrow r + c\rho

(g, h, e_1, e_2)

Special soundness.

Extractor \(K(a_1, a_2, c, z, c^*, z^*)\):

1. \(K_1(a_1, c, z, c^*, z^*) = K_2(a_2, c, z, c^*, z^*)\)
2. \(\rho \leftarrow (z - z^*) / (c - c^*)\)
3. Return \(\rho\)

Analysis. Simple corollary of knowledge of DL protocols. One gets "the same" \(\rho\) from the use of the same \(c\) in both verification equations.
**DDH WITNESS: SHVZK**

\[ L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (h^c, g^c) \text{ for some } c\} \]

**SHVZK Simulator** \( S(c) \):

1. \( z \leftarrow \mathbb{Z}_p \)
2. \( (a_1, a_2) \leftarrow (h^z / e_1^c, g^z / e_2^c) \)
3. Return \( (a_1, a_2; c; z) \)

**Analysis.** Again, simple corollary. Acceptance is tautology. Distribution is the same since in both cases \( c \) and \( z \) are random and \( (a_1, a_2) \) just makes verification accept.
POK: ELGAMAL PLAINTEXT/RANDOMIZER

\[
L = \{(g, h, e_1, e_2), \text{s.t. } (e_1, e_2) = (g^\mu h^\rho, g^\rho) \text{ for some } (\mu, \rho)\}
\]

\[
\begin{align*}
m, r &\leftarrow \mathbb{Z}_p \\
(a_1, a_2) &\leftarrow (g^mr^r, g^r) \\
c &\leftarrow C \\
z_1 &\leftarrow m + c\mu \\
z_2 &\leftarrow r + c\rho
\end{align*}
\]

Accept if \((g^{z_1} h^{z_2}, g^{z_2}) = (a_1e_1^c, a_2e_2^c)\)
REMARKS

- AND-proof gives an easy way to get an AND of two $\Sigma$-protocols
  - but it's better to make sure the result is correct by giving direct security proof
- Example direct security proofs were simple
  - but sometimes it gets very tedious
Assume that $P$ knows a witness to either $x \in L_1$ or $x \in L_2$ and wants to convince $V$ in this

**Important:** $V$ should **not** get to know which witness $P$ knows

**Example:**

- $L_0 = \{ e = \text{Enc}(\text{o}; \varrho) \text{ for some } \varrho \}$, $L_1 = \{ e = \text{Enc}(\text{i}; \varrho) \text{ for some } \varrho \}$

- We know already how to construct protocols for both

- $L = L_0 \cup L_1 = \{ e = \text{Enc}(\text{o}; \varrho) \lor e = \text{Enc}(\text{i}; \varrho) \text{ for some } \varrho \}$

Verifier is convinced plaintext is Boolean
AND VS OR PROOF

- AND proof is simpler:
  - $R_1 \land R_2$ is true iff both $R_1$ and $R_2$ are true
  - Revealing in the proof that $R_i$ is true does not break ZK
- OR proof must not reveal which of $R_1$ and $R_2$ is true
  - another one might or might not be false!
- Additional layer of complexity
QUIZ: "OR" COMPOSITION

For one \( i \in \{1, 2\} \): given protocol \((P_i, V_i)\) for PK \((\omega_i; R_i(x, \omega_i))\)

Goal: construct protocol \((P, V)\) for PK \((\omega; R_1(x, \omega) \lor R_2(x, \omega))\)

Problems:
1. Prover knows only witness \( \omega \) for \( R_i \) and thus cannot execute both branches of OR
2. Verifier should not know which branch \( P \) can execute

Quiz:
* how can \( P \) run the branch, without knowing corresponding secret, such that \( V \) cannot distinguish it from the real run?

Accept if either \( V_1(x, a_1, c_1, z_1) \) or \( V_2(x, a_2, c_2, z_2) \) accepts
ANSWER: USE SIMULATOR

- Need to run protocol without knowing witness?
  - Answer: use the simulator

- Need it to be indistinguishable from real run?
  - Answer: use the simulator
... WITH A TRICK

- **Problem:** $P$ should run one branch non-simulated

- The simulated branch can be created out-of-order, while the “non-simulated" can’t

Recall "special" meant the simulator can start from $c$, then create the rest, while $P$ has to start with $a$

- So **one** of the two must use $c$ provided by $V$

- **Question:** how?
... WITH A TRICK

🔹 Idea: generate $c_i$ first, after seeing verifier's challenge $c$, choose $c_3 - i \leftarrow c - c_i$

🔹 One of $c_1$ and $c_2$ thus must be created "after" seeing $c$

We really need the "special" ZK property
Goal: construct protocol $(P, V)$ for PK $(\omega: R_1(x, \omega) \lor R_2(x, \omega))$
I will not give a full security proof, but it is easy

Completeness: from completeness of first PK, and successful simulation of the second one

Special soundness: OR-extractor runs extractors for both branches. One of them is successful, return this value

SHVZK: since the first PK is SHVZK, and the second one is already simulated
POK: ELGAMAL PLAINTEXT IS BOOLEAN

$L = \{(g, h, e_1, e_2), \text{s.t.} (e_1, e_2) = (g^\mu h^\sigma, g^\sigma) \text{ for some } q \in \mathbb{Z}_p, \mu \in \{0, 1\}\}$

1. \(r \leftarrow \mathbb{Z}_p\)
2. \((a_{11}, a_{12}) \leftarrow (hr, gr)\)
3. \(c_2 \leftarrow C; z_2 \leftarrow \mathbb{Z}_p\)
4. \(a_{21} \leftarrow g^{\mu} h^{z_2} / e_1^{c_2}\)
5. \(a_{22} \leftarrow g^{z_2} / e_2^{c_2}\)

Accept if \(c_1 \in C, (h^{z_1}, g^{z_1}) = (a_{11}e_1^{c_1}, a_{12}e_2^{c_1}), (gh^{z_2}, g^{z_2}) = (a_{21}e_1^{c_2}, a_{22}e_2^{c_2})\)
SECURITY PROOF

Note: there exists a slightly more efficient Boolean proof that does not use simulation
**Σ-PROTOCOLS FOR BOOLEAN CIRCUITS**

- Each circuit can be built from NAND gates
  - \( x \text{ NAND } y = 1 \) iff \( x = 0 \) or \( y = 0 \)
- Easy to verify that NAND is observed:
  - \( x \text{ NAND } y = z \) iff \( x + y + 2z - 2 \in \{0, 1\} \)

**Corollary.** Boolean proofs are sufficient to construct \( Σ \)-protocol for CIRCUIT-SAT

**Proof.** Encrypt each wire value except the last one (which is 1). Prove that each wire value is Boolean. For each gate, prove that NAND is observed
STUDY OUTCOMES

- Σ-protocols for DL-based languages
- Simple examples
- Composition rules
- Everything interesting has a Σ-protocol
NEXT LECTURE

❖ How to build "real ZK" on top of Σ-protocols?

❖ Interactive, four-message ZK
IDEAS FOR HW

- Assume the use of Elgamal

- Construct $\Sigma$-protocol for $PK ((\mu, \rho): c = Enc (\mu; \rho))$
  - “knowledge of plaintext and randomizer”

- For any $k > 0$, construct $\Sigma$-protocol for
  
  $PK ((\mu, \rho): c = Enc (\mu; \rho) \land \mu \in \{0, 1, \ldots, 2^k - 1\})$

- “range proof”. Hint: encrypt $\mu$ bitwise