Efficient Delegation of Zero-Knowledge Proofs of Knowledge in a Pairing-Friendly Setting

Sébastien Canard[1], David Pointcheval[2] and Olivier Sanders[1,2]

(1) Orange Labs, Caen, France
(2) École Normale Supérieure, Paris, France

PKC 2014, March 26, 2014

Agenda

- Zero-Knowledge Proofs of Knowledge
- Delegation of Proofs of Knowledge
- Conclusion

Zero-Knowledge Proofs of Knowledge

Zero-Knowledge Proofs of Knowledge

- Zero-Knowledge Proofs of Knowledge enable a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that:
  - a statement is true.
  - he knows a witness for this fact.

- They must fulfil the following properties:
  - Completeness.
  - Zero-Knowledge: Nothing but the validity of the statement is revealed.
  - Soundness: $\mathcal{P}$ knows a witness.

Schnorr protocol

- Example: the Schnorr protocol for proving knowledge of α such that $V = [\alpha]A$ in a group $\mathbb{G}$ of prime order p.

$$
\begin{array}{lcl}
\mathcal{P} & & \mathcal{V} \\
k \stackrel{\$}{\leftarrow} \mathbb{Z}_p,\ R \leftarrow [k]A & \xrightarrow{\quad R \quad} & \\
 & \xleftarrow{\quad c \quad} & c \leftarrow \{0,1\}^l \\
s \leftarrow k + c \cdot \alpha & \xrightarrow{\quad s \quad} & [s]A \stackrel{?}{=} R + [c]V
\end{array}
$$

Schnorr protocol

- Example: the Schnorr protocol for proving knowledge of α such that $V = [\alpha]A$ in a group $\mathbb{G}$ of prime order p.

$$
\begin{array}{lcl}
\mathcal{P} & & \mathcal{V} \\
k \xleftarrow{\$} \mathbb{Z}_p,\ R \leftarrow [k]A & \xrightarrow{\quad R \quad} & \\
 & \xleftarrow{\quad c \quad} & c \leftarrow \{0,1\}^l \\
s \leftarrow k + c \cdot \alpha & \xrightarrow{\quad s \quad} & [s]A \stackrel{?}{=} R + [c]V
\end{array}
$$

Applications

- These proofs have played a significant role in cryptography:
  - Group Signature
  - E-cash
  - Direct Anonymous Attestation
  - Voting
  - ...
- Indeed, these primitives require to prove that some public elements are well-formed.

Discrete-Log Relation Sets

- Such complex primitives usually deal with a Discrete-Log Relations Set (DLRS, as defined by Kiayias, Tsiounis and Yung):

Relations		Commitments
$V_1 = [\alpha_1]A_{1,1}$	$\longrightarrow$	$R_1 \leftarrow [k_1]A_{1,1}$

Discrete-Log Relation Sets

- Such complex primitives usually deal with a Discrete-Log Relations Set (DLRS, as defined by Kiayias, Tsiounis and Yung):

$$
\begin{array}{cccc}
 & \text{Relations} & & \text{Commitments} \\
\hline
 & V_1 = [\alpha_1]A_{1,1} & \longrightarrow & R_1 \leftarrow [k_1]A_{1,1} \\
\wedge & V_2 = [\alpha_1]A_{2,1} & \longrightarrow & R_2 \leftarrow [k_1]A_{2,1}
\end{array}
$$

Discrete-Log Relation Sets

- Such complex primitives usually deal with a Discrete-Log Relations Set (DLRS, as defined by Kiayias, Tsiounis and Yung):

	Relations		Commitments
	$V_1 = [\alpha_1]A_{1,1}$	$\longrightarrow$	$R_1 \leftarrow [k_1]A_{1,1}$
$\wedge$	$V_2 = [\alpha_1]A_{2,1}$	$\longrightarrow$	$R_2 \leftarrow [k_1]A_{2,1}$
$\wedge$	$V_3 = [\alpha_1]A_{3,1} + [\alpha_2]A_{3,2}$	$\longrightarrow$	$R_3 \leftarrow [k_1]A_{3,1} + [k_2]A_{3,2}$

Discrete-Log Relation Sets

- Such complex primitives usually deal with a Discrete-Log Relations Set (DLRS, as defined by Kiayias, Tsiounis and Yung):

$$
\begin{array}{lllll}
 & \text{Relations} & & \text{Commitments} \\
\hline
 & V_1 = [\alpha_1]A_{1,1} & \longrightarrow & R_1 \leftarrow [k_1]A_{1,1} \\
\wedge & V_2 = [\alpha_1]A_{2,1} & \longrightarrow & R_2 \leftarrow [k_1]A_{2,1} \\
\wedge & V_3 = [\alpha_1]A_{3,1} + [\alpha_2]A_{3,2} & \longrightarrow & R_3 \leftarrow [k_1]A_{3,1} + [k_2]A_{3,2} \\
\wedge & \dots & \longrightarrow & \dots \\
\wedge & V_r = \sum_{j \in \mathcal{I}_r} [\alpha_j]A_{j,r} & \longrightarrow & R_r = \sum_{j \in \mathcal{I}_r} [k_j]A_{j,r}
\end{array}
$$

- The number of commitments grows with the one of relations.

Constrained devices

- The pair (phone/SIM card) is suitable for proving knowledge.
  - The phone is powerful enough for computing the commitments.
  - The secret values can be stored in the SIM card.
- But:
  - The SIM card is not able to compute the commitments.
  - The phone is not fully trusted.

$\Longrightarrow$ How can we delegate these computations?

Methodology

- We split the prover $\mathcal{P}$ into 2 entities:
  - A trusted but constrained one (*e.g.* the SIM card)
  - A more powerful but not fully trusted one (*e.g.* the phone)
- The phone may have access to additional information but cannot recover the secret values.
- The proof must remain zero-knowledge *w.r.t.* the verifier $\mathcal{V}$.

An example: D.A.A.

- A Direct Anonymous Attestation (D.A.A) enables members of a group to anonymously sign on behalf of the group.
- The signer is split into a trusted entity (the TPM) and a not fully trusted one (the Host):
  - Anonymity *w.r.t* the Host is not required.
  - Non-frameability is required.
- The Host can have access to the member's certificate but not to his secret key.

Delegation of Proofs of Knowledge

Bilinear groups

- Most efficient implementations of the previous primitives use bilinear groups.
- Bilinear groups are a set of 3 groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a map e such that:

$$\forall (X, \widetilde{X}) \in \mathbb{G}_1 \times \mathbb{G}_2 \text{ and } a, b \in \mathbb{Z}_p \; e([a]X, [b]\widetilde{X}) = e(X, \widetilde{X})^{a \cdot b}$$
$$\forall (X_1, X_2) \in \mathbb{G}_1^2, e(X_1 + X_2, \widetilde{X}) = e(X_1, \widetilde{X}) \cdot e(X_2, \widetilde{X})$$

A first Step

- To prove knowledge of α such that :

$$V_1 = [\alpha]A_1, V_2 = [\alpha]A_2, ..., V_n = [\alpha]A_n$$
$$\text{with } A_i \in \mathbb{G}_1$$

- We can compute the commitment in $\mathbb{G}_2$:

$$\left.\begin{array}{l} R_1 \leftarrow [k]A_1 \\ R_2 \leftarrow [k]A_2 \\ ... \\ R_n \leftarrow [k]A_n \end{array}\right\} \Longrightarrow \widetilde{R} \leftarrow [k]\widetilde{G}, \text{ for some } \widetilde{G} \in \mathbb{G}_2$$

- Transmit c and $s = k + c \cdot \alpha$ as in the Schnorr protocol.
- And verify it in $\mathbb{G}_T$, for all $1 \le i \le n$:

$$e([s]A_i, \widetilde{G}) \stackrel{?}{=} e(A_i, \widetilde{R}) \cdot e(V_i, \widetilde{G})^c$$

A first Step

- The SIM card only has to compute one scalar multiplication, instead of n.
- The verification now involves pairings but in many cases the verifier will be able to perform them quickly.
- The proof is sound, but not zero-knowledge!
  - From $\widetilde{R}$ we can recover $[\alpha]\widetilde{G} \Rightarrow$ it cannot be sent to $\mathcal{V}$.
  - From $[\alpha]\widetilde{G}$ we cannot recover $\alpha \Rightarrow$ it can be sent to the phone.
- D.A.A. Example: Knowledge of $[\alpha]\widetilde{G}$ does not allow the Host to impersonate the TPM.
  $\Longrightarrow$ Security of the scheme is ensured.

Making the proof Zero-Knowledge

- To make the proof zero-knowledge, the phone will bind $\widetilde{R}$ to each A_i:

$$\forall 1 \leq i \leq n : b_i \xleftarrow{\$} \mathbb{Z}_p,\ B_i \leftarrow [b_i^{-1}]A_i \text{ and } \widetilde{B}_i \leftarrow [b_i]\widetilde{R}$$

- $(B_i, \widetilde{B}_i)$ are sent to $\mathcal{V}$ which can check the proof:

$$e([s]A_i, \widetilde{G}) \stackrel{?}{=} e(B_i, \widetilde{B}_i) \cdot e(V_i, \widetilde{G})^c$$

- The proof is now zero-knowledge but we must extend it to more complex relations:

$$V = \sum_{j=1}^{m} [\alpha_j]A_j$$

A first protocol

- To remain zero-knowledge, the phone must bind the different commitments $\widetilde{R}_j \leftarrow [k_j]\widetilde{G}$.
- If we knew the elements $\widetilde{A}_j \leftarrow [\prod_{k \neq j} a_k]\widetilde{G}$ where $A_j = [a_j]G$, the phone could:
  - select $t_1, \ldots, t_{m-1} \xleftarrow{\$} \mathbb{Z}_p$ and $t_m \in \mathbb{Z}_p$ such that $\sum_{j=1}^{m} t_j = 0$.
  - compute and send $B_j \leftarrow [b_j^{-1}]A_j$ and $\widetilde{B}_j \leftarrow [b_j](\widetilde{R}_j + [t_j]\widetilde{A}_j)$
- $\mathcal{V}$ could check that:

$$e(\sum_{j=1}^{m}[s_j]A_j, \widetilde{G}) \stackrel{?}{=} e(V, \widetilde{G})^c \cdot \prod_{j=1}^{m} e(B_j, \widetilde{B}_j)$$

A second protocol

- Knowledge of $\widetilde{A}_j$ is a strong assumption but:
  - If $m = 1$, $\widetilde{A}_j = \widetilde{G}$
  - If $m = 2$ then $\{\widetilde{A}_j\}_j = \{A_j\}_j$ when using a symmetric pairing.
- We need to modify this solution to suit the other cases. The phone:
  - selects $t_1, \ldots t_m \xleftarrow{\$} \mathbb{Z}_p$ (without any condition).
  - computes and sends $H \leftarrow \sum_{j=1}^{m} [t_j] A_j$, B_j and $\widetilde{B}_j \leftarrow [b_j](\widetilde{R}_j + [t_j]\widetilde{G})$
- Verification is similar:

$$e(H + \sum_{j=1}^{m} [s_j] A_j, \widetilde{G}) \stackrel{?}{=} e(V, \widetilde{G})^c \cdot \prod_{j=1}^{m} e(B_j, \widetilde{B}_j)$$

In summary

- For a relation:

$$V = \sum_{j=1}^{m} [\alpha_j] A_j$$

- The SIM card computes:

$$\widetilde{R}_j \leftarrow [k_j]\widetilde{G}$$

- The commitments received by $\mathcal{V}$ are:

$$H \leftarrow \sum_{j=1}^{m} [t_j] A_j, \; B_j \leftarrow [b_j^{-1}] A_j \text{ and } \widetilde{B}_j \leftarrow [b_j](\widetilde{R}_j + [t_j]\widetilde{G})$$

In summary

- For a relation:

$$V = \sum_{j=1}^{m} [\alpha_j] A_j$$

- The SIM card computes:

$$\widetilde{R}_j \leftarrow [k_j] \widetilde{G}$$

- The commitments received by $\mathcal{V}$ are:

$$H \leftarrow \sum_{j=1}^{m} [t_j] A_j, \; B_j \leftarrow [b_j^{-1}] A_j \text{ and } \widetilde{B}_j \leftarrow [b_j](\widetilde{R}_j + [t_j]\widetilde{G})$$

- The factors $(b_j)_j$ bind the elements $\widetilde{R}_j$ to the basis $(A_j)_j$.

$\Longrightarrow$ else, $\mathcal{V}$ would learn $[\alpha_j]\widetilde{G}$

In summary

- For a relation:

$$V = \sum_{j=1}^{m} [\alpha_j] A_j$$

- The SIM card computes:

$$\widetilde{R}_j \leftarrow [k_j] \widetilde{G}$$

- The commitments received by $\mathcal{V}$ are:

$$H \leftarrow \sum_{j=1}^{m} [t_j] A_j, \; B_j \leftarrow [b_j^{-1}] A_j \text{ and } \widetilde{B}_j \leftarrow [b_j](\widetilde{R}_j + [t_j]\widetilde{G})$$

- The factors $(t_j)_j$ bind the elements $\widetilde{R}_j$ together.

$\Longrightarrow$ else, $\mathcal{V}$ would learn $e(A_j, \widetilde{G})^{\alpha_j}$

In summary

- For a relation:

$$V = \sum_{j=1}^{m} [\alpha_j] A_j$$

- The SIM card computes:

$$\widetilde{R}_j \leftarrow [k_j]\widetilde{G}$$

- The commitments received by $\mathcal{V}$ are:

$$H \leftarrow \sum_{j=1}^{m} [t_j] A_j,\ B_j \leftarrow [b_j^{-1}] A_j \text{ and } \widetilde{B}_j \leftarrow [b_j](\widetilde{R}_j + [t_j]\widetilde{G})$$

- These additional factors must be cancelled.

  $\Longrightarrow$ else, $\mathcal{V}$ could not check the validity of the proof.

Security

- The proof is complete.
- The proof is sound.
- The proof is zero-knowledge *w.r.t.* $\mathcal{V}$.
- The proof only leaks $[\alpha_1]\widetilde{G}, \ldots, [\alpha_m]\widetilde{G}$ to the phone.

Complexity

- To prove knowledge of $\alpha_1, ..., \alpha_m$ such that:

$$V_1 = [\alpha_1]A_{1,1} + ... + [\alpha_m]A_{1,m}$$
$$...$$
$$V_n = [\alpha_1]A_{n,1} + ... + [\alpha_m]A_{n,m}$$

- The SIM card must perform:
  - $n \times m$ scalar multiplications with the Schnorr protocol.
  - m scalar multiplications with our protocol.

- $\widetilde{G}$ is a random element from $\mathbb{G}_2 \Longrightarrow$ each $\widetilde{R}_j$ can be pre-computed.

- Each $\widetilde{R}_j$ is sent to the phone $\Longrightarrow$ The SIM card just needs to store the seed and the index used to generate the factors k_j.

Complexity

- The work is shifted to the phone and to the verifier:
  - For the phone: between $2n \times m$ and $4n \times m$ scalar multiplications (half of them being pre-computable).
  - For $\mathcal{V}$: $n \times (m+1)$ pairing computations.
- This tradeoff is motivated by the different computational powers:
  - SIM card / Phone
  - SIM card / Server (acting as $\mathcal{V}$)

Conclusion

Conclusion

- Our protocols are zero-knowledge proofs of knowledge of a DLRS.
- The prover $\mathcal{P}$ is split between two entities.
- The low-power entity only has to pre-compute one scalar multiplication by secret.
- The protocol only leaks few information to the delegatee.
- It involves additional computations (compared to the Schnorr protocol) for the delegatee and $\mathcal{V}$.

thank you