Les preuves sans divulgations de connaissances

Fabrice Mouhartem

19/11/2014

ENS DE LYON

Outline

Outline

Le problème de l'identification

Problème : On veut prouver à quelqu'un que l'on est le seul à posséder un secret.

Le problème de l'identification

Problème : On veut prouver à quelqu'un que l'on est le seul à posséder un secret.

Solutions ?

1. On peut envoyer son secret ?

Le problème de l'identification

Problème : On veut prouver à quelqu'un que l'on est le seul à posséder un secret.

Solutions ?

1. On peut envoyer son secret ?
2. On peut utiliser sa clé privée comme secret ?

Le problème de l'identification

Problème : On veut prouver à quelqu'un que l'on est le seul à posséder un secret.

Solutions ?

1. On peut envoyer son secret ?
2. On peut utiliser sa clé privée comme secret ?

Ces solutions ne sont pas satisfaisantes.

L'exemple de Quisquater

Au tableau

Pour plus d'explication : `https://en.wikipedia.org/wiki/Zero-knowledge_proof#Abstract_example`

Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. On a un secret $s \in \mathbb{Z}_q$.

Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. On a un secret $s \in \mathbb{Z}_q$.
On partage publiquement son secret *caché* : $h := g^s$.

1. On génère $r \leftarrow_R \mathbb{Z}_q$ et on envoie à $a = g^r$ à son challenger

Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. On a un secret $s \in \mathbb{Z}_q$.
On partage publiquement son secret *caché* : $h := g^s$.

1. On génère $r \leftarrow_R \mathbb{Z}_q$ et on envoie à $a = g^r$ à son challenger
2. Le challenger prépare un test $c \in \mathbb{Z}_q$ et me l'envoie

Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. On a un secret $s \in \mathbb{Z}_q$.
On partage publiquement son secret *caché* : $h := g^s$.

1. On génère $r \leftarrow_R \mathbb{Z}_q$ et on envoie à $a = g^r$ à son challenger
2. Le challenger prépare un test $c \in \mathbb{Z}_q$ et me l'envoie
3. On renvoie $z = r + sc$ à son challenger. Il accepte si $g^z = ah^c$

Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. On a un secret $s \in \mathbb{Z}_q$.
On partage publiquement son secret *caché* : $h := g^s$.

1. On génère $r \leftarrow_R \mathbb{Z}_q$ et on envoie à $a = g^r$ à son challenger
2. Le challenger prépare un test $c \in \mathbb{Z}_q$ et me l'envoie
3. On renvoie $z = r + sc$ à son challenger. Il accepte si $g^z = ah^c$

Remarques sur l'exemple

- Si je vois un tel échange, je ne suis pas convaincu

Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. On a un secret $s \in \mathbb{Z}_q$.
On partage publiquement son secret *caché* : $h := g^s$.

1. On génère $r \leftarrow_R \mathbb{Z}_q$ et on envoie à $a = g^r$ à son challenger
2. Le challenger prépare un test $c \in \mathbb{Z}_q$ et me l'envoie
3. On renvoie $z = r + sc$ à son challenger. Il accepte si $g^z = ah^c$

Remarques sur l'exemple

- Si je vois un tel échange, je ne suis pas convaincu
- C'est si dur à inverser une exponentiation ?

Outline

Preuve

Preuve mathématique

En maths une preuve :

- est non interactive
- doit convaincre tous ceux qui la lisent

Preuve

Preuve mathématique

En maths une preuve :

- est non interactive
- doit convaincre tous ceux qui la lisent

Ici on remarque que la preuve est :

- Probabiliste
- Interactive
- Destiné à convaincre uniquement son challenger

Preuve

Preuve mathématique

En maths une preuve :

- est non interactive
- doit convaincre tous ceux qui la lisent

Ici on remarque que la preuve est :

- Probabiliste
- Interactive
- Destiné à convaincre uniquement son challenger

À propos du dernier point : Utile par exemple pour des raisons d'anonymat.

Preuve interactive

Une preuve interactive met en jeu trois composantes :

Preuve interactive

Une preuve interactive met en jeu trois composantes :

1. Un *prouveur* P
2. Un *vérifieur* (ou challenger) V
3. Une relation $R = \{(x, w)\}$ définissant l'ensemble des assertions vraies.

Preuve interactive

Une preuve interactive met en jeu trois composantes :

1. Un *prouveur* P
2. Un *vérifieur* (ou challenger) V
3. Une relation $R = \{(x, w)\}$ définissant l'ensemble des assertions vraies. On peut voir x comme une instance d'un problème et w comme une réponse (ou un témoin) à ce problème.

Preuve interactive

Propriétés

Consistance : Si P et V suivent le protocole de la preuve sur l'entrée x et P connaissant w avec $(x, w) \in R$, alors V accepte toujours.
Robustesse : Si $(x, w) \notin R$, alors aucun prouveur P^* ne peut convaincre V avec probabilité non négligeable.

Logarithme discret

DLP

Étant donné (G, g, h) avec $G = \langle g \rangle$ cyclique, et $h = g^y$, on veut calculer y.

Logarithme discret

DLP

Étant donné (G, g, h) avec $G = \langle g \rangle$ cyclique, et $h = g^y$, on veut calculer y.

Modèle pour un groupe générique d'ordre q : on peut multiplier, avoir le neutre et inverser en une opération, et tester gratuitement l'égalité.

Logarithme discret

DLP

Étant donné (G, g, h) avec $G = \langle g \rangle$ cyclique, et $h = g^y$, on veut calculer y.

Modèle pour un groupe générique d'ordre q : on peut multiplier, avoir le neutre et inverser en une opération, et tester gratuitement l'égalité.

Tout ce qu'on peut faire c'est générer une suite $(a_k, b_k)_k$ et calculer l'ensemble $\left\{g^{a_k} h^{b_k}\right\}_k$ jusqu'à tomber sur une collision qui révèlera le logarithme discret.

Logarithme discret

DLP

Étant donné (G, g, h) avec $G = \langle g \rangle$ cyclique, et $h = g^y$, on veut calculer y.

Modèle pour un groupe générique d'ordre q : on peut multiplier, avoir le neutre et inverser en une opération, et tester gratuitement l'égalité.

Tout ce qu'on peut faire c'est générer une suite $(a_k, b_k)_k$ et calculer l'ensemble $\left\{g^{a_k} h^{b_k}\right\}_k$ jusqu'à tomber sur une collision qui révèlera le logarithme discret.La probabilité de tomber sur une collision est quadratique en le nombre d'essai.

Logarithme discret

DLP

Étant donné (G, g, h) avec $G = \langle g \rangle$ cyclique, et $h = g^y$, on veut calculer y.

Modèle pour un groupe générique d'ordre q : on peut multiplier, avoir le neutre et inverser en une opération, et tester gratuitement l'égalité.

Tout ce qu'on peut faire c'est générer une suite $(a_k, b_k)_k$ et calculer l'ensemble $\left\{g^{a_k} h^{b_k}\right\}_k$ jusqu'à tomber sur une collision qui révèlera le logarithme discret.La probabilité de tomber sur une collision est quadratique en le nombre d'essai. On a donc besoin de $\mathcal{O}(\sqrt{q})$ essais pour avoir une probabilité suffisante de faire tomber le logarithme discret.

Logarithme discret

DLP

Étant donné (G, g, h) avec $G = \langle g \rangle$ cyclique, et $h = g^y$, on veut calculer y.

Modèle pour un groupe générique d'ordre q : on peut multiplier, avoir le neutre et inverser en une opération, et tester gratuitement l'égalité.

Tout ce qu'on peut faire c'est générer une suite $(a_k, b_k)_k$ et calculer l'ensemble $\left\{g^{a_k} h^{b_k}\right\}_k$ jusqu'à tomber sur une collision qui révèlera le logarithme discret.La probabilité de tomber sur une collision est quadratique en le nombre d'essai. On a donc besoin de $\mathcal{O}(\sqrt{q})$ essais pour avoir une probabilité suffisante de faire tomber le logarithme discret.
Remarque : Dans la pratique on ne travaille pas sur un groupe générique.

Outline

Rappel : Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. Le prouveur a un secret $s \in \mathbb{Z}_q$.
Le prouveur partage publiquement son secret *caché* : $h := g^s$.

1. Le prouveur génère $r \leftarrow_R \mathbb{Z}_q$ et on envoie à $a = g^r$ à son challenger
2. Le challenger prépare un test $c \in \mathbb{Z}_q$ et l'envoie au prouveur
3. Le prouveur renvoie $z = r + sc$ à son challenger. Il accepte si $g^z = ah^c$

Remarques : Ici $R = \{(g^w, w)\}_w$ l'ensemble instances du problème du logarithme discret.

Rappel : Le protocole de Schnorr

Soit un groupe cyclique $G = \langle g \rangle$ d'ordre q. Le prouveur a un secret $s \in \mathbb{Z}_q$.
Le prouveur partage publiquement son secret *caché* : $h := g^s$.

1. Le prouveur génère $r \leftarrow_R \mathbb{Z}_q$ et on envoie à $a = g^r$ à son challenger
2. Le challenger prépare un test $c \in \mathbb{Z}_q$ et l'envoie au prouveur
3. Le prouveur renvoie $z = r + sc$ à son challenger. Il accepte si $g^z = ah^c$

Remarques : Ici $R = \{(g^w, w)\}_w$ l'ensemble instances du problème du logarithme discret.
On peut généraliser ce schéma.

Protocole Σ

Soit $(x, w) \in R$, x est partagée publiquement et P possède secrètement w.

1. P envoie un message a à V
2. V envoie un challenge c à P
3. P répond à V par z, qui décide alors d'accepter à partir des données observées pendant l'échange : a, c, z et x

Protocole Σ

Soit $(x, w) \in R$, x est partagée publiquement et P possède secrètement w.

1. P envoie un message a à V
2. V envoie un challenge c à P
3. P répond à V par z, qui décide alors d'accepter à partir des données observées pendant l'échange : a, c, z et x

On peut retranscrire la conversation par la donnée du triplet (a, c, z)

Protocole Σ

On veut vérifier les propriétés suivantes :
Cohérence : Si P et V suivent le protocole sur l'entrée $(x, w) \in R$, alors V accepte.

Protocole Σ

On veut vérifier les propriétés suivantes :
Cohérence : Si P et V suivent le protocole sur l'entrée $(x, w) \in R$, alors V accepte.
Robustesse spéciale : Il existe un algorithme *efficace* $\mathcal{A}$ qui étant donné x et deux transcriptions (a, c, z) et (a, c', z') peut reconstruire un w tel que $(x, w) \in R$.

Protocole Σ

On veut vérifier les propriétés suivantes :
Cohérence : Si P et V suivent le protocole sur l'entrée $(x, w) \in R$, alors V accepte.
Robustesse spéciale : Il existe un algorithme *efficace* $\mathcal{A}$ qui étant donné x et deux transcriptions (a, c, z) et (a, c', z') peut reconstruire un w tel que $(x, w) \in R$.
Zéro-divulgation de connaissance avec un vérifieur honnête : Il existe un simulateur *efficace* $\mathcal{M}$ qui étant donné une entrée x, c renvoie une transcription (a, c, z) indistinguable en distribution d'un vrai échange.

Protocole Σ

On veut vérifier les propriétés suivantes :
Cohérence : Si P et V suivent le protocole sur l'entrée $(x, w) \in R$, alors V accepte.
Robustesse spéciale : Il existe un algorithme *efficace* $\mathcal{A}$ qui étant donné x et deux transcriptions (a, c, z) et (a, c', z') peut reconstruire un w tel que $(x, w) \in R$.
Zéro-divulgation de connaissance avec un vérifieur honnête : Il existe un simulateur *efficace* $\mathcal{M}$ qui étant donné une entrée x, c renvoie une transcription (a, c, z) indistinguable en distribution d'un vrai échange.
Remarque : Dans la pratique un vérifieur honnête n'est pas très utile. Mais c'est un pas vers une preuve *zero-knowledge*.

Preuve zero-knowledge

Idée

On veut garantir que le vérifieur est honnête

Preuve zero-knowledge

Idée

On veut garantir que le vérifieur est honnête

Une solution possible est de l'empêcher de modifier son challenge.

Preuve zero-knowledge

Idée

On veut garantir que le vérifieur est honnête

Une solution possible est de l'empêcher de modifier son challenge. Pour faire ça on va utiliser l'équivalent cryptographique d'une boîte scellée.

Preuve zero-knowledge

Idée

On veut garantir que le vérifieur est honnête

Une solution possible est de l'empêcher de modifier son challenge.
Pour faire ça on va utiliser l'équivalent cryptographique d'une boîte scellée.
On veut donc garantir 2 propriétés :

- Une fois close, on ne peut plus modifier ce qui est dans la boîte
- On ne peut pas savoir ce qu'il y a dans la boîte sans l'ouvrir

Formellement

On dispose de 3 algorithmes :
Setup : qui nous renvoie une clé publique pk
Com$_{pk}(m)$ **:** qui renvoie la boite close com et sa clé dec
Open$_{pk}(com, dec)$ **:** qui renvoie une preuve que m est bien le message caché par com

Formellement

On dispose de 3 algorithmes :
Setup : qui nous renvoie une clé publique pk
Com$_{pk}(m)$ **:** qui renvoie la boite close com et sa clé dec
Open$_{pk}(com, dec)$ **:** qui renvoie une preuve que m est bien le message caché par com
Digression – Ça existe vraiment : Protocole de Pedersen

Preuve zero-knowledge

Comment utiliser cette boîte scellée ?

Preuve zero-knowledge

Comment utiliser cette boîte scellée ?

1. V crée un challenge c et envoie son commit à P
2. P envoie un message a à V
3. V envoie le *decommitment* de c à V
4. P répond à V par z, qui décide alors d'accepter à partir des données observées pendant l'échange

Preuve zero-knowledge

Comment utiliser cette boîte scellée ?

1. V crée un challenge c et envoie son commit à P
2. P envoie un message a à V
3. V envoie le *decommitment* de c à V
4. P répond à V par z, qui décide alors d'accepter à partir des données observées pendant l'échange

Ainsi V ne connaît pas a au moment de construire son challenge, et P ne connaît pas le challenge au moment d'envoyer son premier message.

Outline

Preuves OU

Supposons qu'on ait deux propositions P_1 et P_2 et un protocole Σ pour chacune d'entre elles.

Preuves OU

Supposons qu'on ait deux propositions P_1 et P_2 et un protocole Σ pour chacune d'entre elles.
Construire un protocole pour $P_1 \wedge P_2$ est trivial.

Preuves OU

Supposons qu'on ait deux propositions P_1 et P_2 et un protocole Σ pour chacune d'entre elles.
Construire un protocole pour $P_1 \wedge P_2$ est trivial.
Comment construire une preuve pour $P_1 \vee P_2$ sans pour autant qu'on sache si on a prouvé P_1 ou P_2 ?

Preuves OU

Ici l'idée va être de partager le challenge en deux entre la vraie preuve et la potentiellement fausse preuve.

Preuves OU

Ici l'idée va être de partager le challenge en deux entre la vraie preuve et la potentiellement fausse preuve.
On suppose connaître $(x_b, w_b) \in R$ (mais on n'a pas w_{1-b}).

1. On génère un challenge aléatoire c_{1-b} et on utilise un simulateur pour obtenir un transcript $(a_{1-b}, c_{1-b}, z_{1-b})$
2. On choisit a_b et on envoie (a_0, a_1) à V
3. V nous renvoie un challenge c
4. On calcule $c_b = c \oplus c_{1-b}$ et on génère z_b à partir de c_b. On envoie (c_0, c_1) et (z_0, z_1)
5. V accepte si les deux transcripts (a_0, c_0, z_0) et (a_1, c_1, z_1) sont valides et si $c = c_0 \oplus c_1$

Remarques

- Si on ne connaît pas au moins un des deux témoins, on ne pourra pas répondre au challenge c_b

Remarques

- Si on ne connaît pas au moins un des deux témoins, on ne pourra pas répondre au challenge c_b
- c_0 et c_1 sont identiquement distribués si le challenge est vraiment aléatoire

Remarques

- Si on ne connaît pas au moins un des deux témoins, on ne pourra pas répondre au challenge c_b
- c_0 et c_1 sont identiquement distribués si le challenge est vraiment aléatoire

Applications

- Authentification anonyme : au lieu de dire : « je connais le w d'un g^w » on peut prouver « je connais un w_{i_0} d'un g^{w_i} parmi un ensemble fini »

Remarques

- Si on ne connaît pas au moins un des deux témoins, on ne pourra pas répondre au challenge c_b
- c_0 et c_1 sont identiquement distribués si le challenge est vraiment aléatoire

Applications

- Authentification anonyme : au lieu de dire : « je connais le w d'un g^w » on peut prouver « je connais un w_{i_0} d'un g^{w_i} parmi un ensemble fini »
- Élection : « ceci est un vote pour le candidat A ou B ou C »

Remarques

- Si on ne connaît pas au moins un des deux témoins, on ne pourra pas répondre au challenge c_b
- c_0 et c_1 sont identiquement distribués si le challenge est vraiment aléatoire

Applications

- Authentification anonyme : au lieu de dire : « je connais le w d'un g^w » on peut prouver « je connais un w_{i_0} d'un g^{w_i} parmi un ensemble fini »
- Élection : « ceci est un vote pour le candidat A ou B ou C »
- Je veux accéder à une base de donnée sans qu'elle sache quelles requêtes ont été demandées

Signature numérique

Étant donné un message, on veut pouvoir dire « j'ai écrit ce message ».

Signature numérique

Étant donné un message, on veut pouvoir dire « j'ai écrit ce message ».
De façon à avoir les propriétés suivantes :

- On peut vérifier que j'ai bien signé le message
- On peut vérifier que le message n'a pas été altéré
- Une autre personne que moi ne devrait pas être capable de signer mes messages
- Une signature pour un message ne peut être utilisée pour signer un autre message
- Je ne peux pas nier avoir signé un message si je l'ai fait

Signature numérique

Problème

La preuve est interactive.

Signature numérique

Problème

La preuve est interactive.

Solution

On fait comme s'il y avait un challenger.

Si on a un oracle aléatoire $\mathcal{H}$, on génère la preuve $(a, H(a, x), z)$.

Signature numérique

Problème

La preuve est interactive.

Solution

On fait comme s'il y avait un challenger.

Si on a un oracle aléatoire $\mathcal{H}$, on génère la preuve $(a, H(a, x), z)$.

Pour la signature, il suffit d'injecter le message dans la preuve : $(a, H(a, (x, m)), z)$.

Outline

Conclusion

- On peut produire une preuve sans qu'un vérifieur même malicieux puisse dériver l'information secrète
- On ne peut pas convaincre quelqu'un d'autre
- Il s'agit d'un outil puissant pour résoudre ce genre de problèmes et assez efficace (quelques exponentiations pour le protocole de Schnorr)
- Il s'agit aussi d'un outil qui peut être utilisé dans d'autres questions de cryptologie pour répondre aux problèmes de « garantie »

Références

 C. Hazay & Y. Lindell, *Sigma Protocols and Efficient Zero-Knowledge*. Springer 2010

 I. Damgård, *On Σ-protocols*. 2010

Merci de votre attention :)