

# The computational and decisional Diffie-Hellman assumptions in CryptoVerif

Bruno Blanchet and David Pointcheval

CNRS, École Normale Supérieure, INRIA, Paris

July 2010

# Motivation

CryptoVerif is a prover for security protocols that

- is sound in the **computational model**
- produces proofs by **sequences of games**
- can give **asymptotic** or **exact security** results
- provides a **generic** method for specifying assumptions on **cryptographic primitives**

Our goal: extend CryptoVerif to **Diffie-Hellman key agreements**.

- an important primitive;
- difficult for handle in formal protocol provers.

# Outline

- 1 Decisional Diffie-Hellman (DDH) assumption, basic model.
- 2 Computational Diffie-Hellman (CDH) assumption, basic model.
- 3 Why this is not enough for protocols relying on Diffie-Hellman key agreements.
- 4 Computational Diffie-Hellman (CDH) assumption, extended model.
- 5 Decisional Diffie-Hellman (DDH) assumption, extended model.

# Decisional Diffie-Hellman assumption

Consider a multiplicative cyclic group  $G$  of order  $q$ , with generator  $g$ . A probabilistic polynomial-time adversary has a negligible probability of distinguishing

$$(g^a, g^b, g^{ab}) \text{ for random } a, b \in \mathbb{Z}_q^*$$

$$\text{and } (g^a, g^b, g^c) \text{ for random } a, b, c \in \mathbb{Z}_q^*$$

# Decisional Diffie-Hellman assumption in CryptoVerif

Consider a multiplicative cyclic group  $G$  of order  $q$ , with generator  $g$ .  
 A probabilistic polynomial-time adversary has a negligible probability of distinguishing

$$(g^a, g^b, g^{ab}) \text{ for random } a, b \in \mathbb{Z}_q^*$$

$$\text{and } (g^a, g^b, g^c) \text{ for random } a, b, c \in \mathbb{Z}_q^*$$

In CryptoVerif,

$$!^{i \leq N} \text{ new } a : Z; \text{ new } b : Z;$$

$$(OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), ODH() := \text{exp}(g, \text{mult}(a, b)))$$

$$\approx$$

$$!^{i \leq N} \text{ new } a : Z; \text{ new } b : Z; \text{ new } c : Z;$$

$$(OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), ODH() := \text{exp}(g, c))$$

# Decisional Diffie-Hellman assumption in CryptoVerif

$\forall i \leq N$  **new**  $a : Z$ ; **new**  $b : Z$ ;  
 $(OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), ODH() := \text{exp}(g, \text{mult}(a, b)))$

$\approx$

$\forall i \leq N$  **new**  $a : Z$ ; **new**  $b : Z$ ; **new**  $c : Z$ ;  
 $(OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), ODH() := \text{exp}(g, c))$

We replace  $g^{ab}$  with  $g^c$  for some fresh random number  $c$ , provided  $a$  and  $b$  are random numbers used only in  $g^a$ ,  $g^b$ , and  $g^{ab}$ .

Application: semantic security of **El Gamal** (A. Chaudhuri).

# Computational Diffie-Hellman assumption

Consider a multiplicative cyclic group  $G$  of order  $q$ , with generator  $g$ . A probabilistic polynomial-time adversary has a negligible probability of **computing**  $g^{ab}$  from  $g$ ,  $g^a$ ,  $g^b$ , for random  $a, b \in \mathbb{Z}_q^*$ .

# Computational Diffie-Hellman assumption in CryptoVerif

Consider a multiplicative cyclic group  $G$  of order  $q$ , with generator  $g$ . A probabilistic polynomial-time adversary has a negligible probability of **computing**  $g^{ab}$  from  $g$ ,  $g^a$ ,  $g^b$ , for random  $a, b \in \mathbb{Z}_q^*$ .

In CryptoVerif, this can be written

$$\begin{aligned}
 & !^{i \leq N} \text{new } a : Z; \text{new } b : Z; \\
 & \quad (OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), \\
 & \quad \quad !^{i' \leq N'} \text{OCDH}(z : G) := z = \text{exp}(g, \text{mult}(a, b))) \\
 & \approx \\
 & !^{i \leq N} \text{new } a : Z; \text{new } b : Z; \\
 & \quad (OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), \\
 & \quad \quad !^{i' \leq N'} \text{OCDH}(z : G) := \text{false})
 \end{aligned}$$

# Computational Diffie-Hellman assumption in CryptoVerif

Consider a multiplicative cyclic group  $G$  of order  $q$ , with generator  $g$ . A probabilistic polynomial-time adversary has a negligible probability of **computing**  $g^{ab}$  from  $g$ ,  $g^a$ ,  $g^b$ , for random  $a, b \in \mathbb{Z}_q^*$ .

In CryptoVerif, this can be written

$$\begin{aligned}
 & !^{i \leq N} \text{new } a : Z; \text{new } b : Z; \\
 & \quad (OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), \\
 & \quad !^{i' \leq N'} \text{OCDH}(z : G) := z = \text{exp}(g, \text{mult}(a, b))) \\
 & \approx \\
 & !^{i \leq N} \text{new } a : Z; \text{new } b : Z; \\
 & \quad (OA() := \text{exp}(g, a), OB() := \text{exp}(g, b), \\
 & \quad !^{i' \leq N'} \text{OCDH}(z : G) := \text{false})
 \end{aligned}$$

Application: semantic security of **hashed El Gamal in the random oracle model** (A. Chaudhuri).

# Typical protocol using the Diffie-Hellman key agreement

Assumptions on primitives:

- CDH +  $h$  is a hash function in the random oracle model
- or DDH +  $h$  is an entropy extractor

A simplified form of a Diffie-Hellman key agreement protocol:

Message 1.  $A \rightarrow B$ :  $g^a$  for random  $a$

Message 2.  $B \rightarrow A$ :  $g^b$  for random  $b$

The shared key is  $h(g^{ab}) = h((g^a)^b) = h((g^b)^a)$

(Signatures omitted for simplicity.)

# Typical protocol using the Diffie-Hellman key agreement in CryptoVerif

$!^{iA \leq N} cA(); \mathbf{new} a : Z; \overline{cA} \langle \exp(g, a) \rangle; cA(gb). \mathbf{let} k = h(\exp(gb, a)) \mathbf{in} \dots$

|

$!^{iB \leq N} cB(); \mathbf{new} b : Z; \overline{cB} \langle \exp(g, b) \rangle; cA(ga). \mathbf{let} k = h(\exp(ga, b)) \mathbf{in} \dots$

|

$!^{iH \leq nH} cH(x); \overline{cH} \langle h(x) \rangle$

Cannot be transformed by the previous CDH/DDH equivalences, because  $a$  and  $b$  are chosen in parallel processes, not one after the other under the same replication.

# Extending the formalization of CDH in CryptoVerif

After applying the security assumption on the hash function  $h$ ,

- $h(x)$  returns a fresh random number if  $h(x)$  has not already been called,
- and the same result as the previous call otherwise.

Hence  $h(x)$  is replaced with lookups that **compare  $x$  with the other arguments of  $h$** .

$$!^{iA \leq N} cA(); \text{ new } a : Z; \overline{cA} \langle \exp(g, a) \rangle; cA(gb) \dots \exp(gb[u], a[u]) = \exp(gb, a) \\ \dots \exp(ga[u'], b[u']) = \exp(gb, a) \dots x[u''] = \exp(gb, a) \dots$$

$$!^{iB \leq N} cB(); \text{ new } b : Z; \overline{cB} \langle \exp(g, b) \rangle; cA(ga) \dots \exp(gb[u], a[u]) = \exp(ga, b) \\ \dots \exp(ga[u'], b[u']) = \exp(ga, b) \dots x[u''] = \exp(ga, b) \dots$$

$$!^{iH \leq nH} cH(x); \dots \exp(gb[u], a[u]) = x \dots \exp(ga[u'], b[u']) = x \dots x[u''] = x.$$

# Extending the formalization of CDH in CryptoVerif

```

!ia≤na new a : Z; (OA() := exp(g, a), Oa() := a,
  !iaCDH≤naCDH OCDHa(m : G, j ≤ nb) := m = exp(g, mult(b[j], a))),
!ib≤nb new b : Z; (OB() := exp(g, b), Ob() := b,
  !ibCDH≤nbCDH OCDHb(m : G, j ≤ na) := m = exp(g, mult(a[j], b)))
≈
!ia≤na new a : Z; (OA() := exp(g, a), Oa() := a,
  !iaCDH≤naCDH OCDHa(m : G, j ≤ nb) :=
    if Ob[j] or Oa has been called then
      m = exp(g, mult(b[j], a))
    else false),
!ib≤nb new b : Z; (OB() := exp(g, b), Ob() := b,
  !ibCDH≤nbCDH OCDHb(m : G, j ≤ na) := (symmetric of OCDHa))

```

# Extending the formalization of CDH in CryptoVerif

$$\begin{aligned}
 & !^{ia \leq na} \text{ new } a : Z; (OA() := \exp(g, a), Oa() := a, \\
 & \quad !^{iaCDH \leq naCDH} OCDHa(m : G, j \leq nb) := m = \exp(g, \text{mult}(b[j], a))), \\
 & !^{ib \leq nb} \text{ new } b : Z; (OB() := \exp(g, b), Ob() := b, \\
 & \quad !^{ibCDH \leq nbCDH} OCDHb(m : G, j \leq na) := m = \exp(g, \text{mult}(a[j], b))) \\
 & \approx \\
 & !^{ia \leq na} \text{ new } a : Z; (OA() := \exp(g, a), Oa() := \text{let } ka = \text{mark in } a, \\
 & \quad !^{iaCDH \leq naCDH} OCDHa(m : G, j \leq nb) := \\
 & \quad \quad \text{find } u \leq nb \text{ suchthat defined}(kb[u], b[u]) \wedge b[j] = b[u] \text{ then} \\
 & \quad \quad \quad m = \exp(g, \text{mult}(b[j], a)) \\
 & \quad \quad \text{else if defined}(ka) \text{ then } m = \exp(g, \text{mult}(b[j], a)) \text{ else false}), \\
 & !^{ib \leq nb} \text{ new } b : Z; (OB() := \exp(g, b), Ob() := \text{let } kb = \text{mark in } b, \\
 & \quad !^{ibCDH \leq nbCDH} OCDHb(m : G, j \leq na) := (\text{symmetric of } OCDHa))
 \end{aligned}$$

# Extending the formalization of CDH in CryptoVerif

$\!|^{ia \leq na}$  **new**  $a : Z$ ; ( $OA() := \text{exp}(g, a)$ ,  $Oa()[3] := a$ ,

$\!|^{iaCDH \leq naCDH}$   $OCDHa(m : G, j \leq nb)$  [required]  $:= m = \text{exp}(g, \text{mult}(b[j],$

$\!|^{ib \leq nb}$  **new**  $b : Z$ ; ( $OB() := \text{exp}(g, b)$ ,  $Ob()[3] := b$ ,

$\!|^{ibCDH \leq nbCDH}$   $OCDHb(m : G, j \leq na) := m = \text{exp}(g, \text{mult}(a[j], b)))$

$\approx$   $(\#OCDHa + \#OCDHb) \times \max(1, e^2 \#Oa) \times \max(1, e^2 \#Ob) \times$   
 $pCDH(\text{time} + (na + nb + \#OCDHa + \#OCDHb) \times \text{time}(\text{exp}))$

$\!|^{ia \leq na}$  **new**  $a : Z$ ; ( $OA() := \text{exp}'(g, a)$ ,  $Oa() := \text{let } ka = \text{mark in } a$ ,

$\!|^{iaCDH \leq naCDH}$   $OCDHa(m : G, j \leq nb) :=$

**find**  $u \leq nb$  **suchthat**  $\text{defined}(kb[u], b[u]) \wedge b[j] = b[u]$  **then**

$m = \text{exp}(g, \text{mult}(b[j], a))$

**else if**  $\text{defined}(ka)$  **then**  $m = \text{exp}'(g, \text{mult}(b[j], a))$  **else false**,

$\!|^{ib \leq nb}$  **new**  $b : Z$ ; ( $OB() := \text{exp}'(g, b)$ ,  $Ob() := \text{let } kb = \text{mark in } b$ ,

$\!|^{ibCDH \leq nbCDH}$   $OCDHb(m : G, j \leq na) := (\text{symmetric of } OCDHa)$

# Other declarations for Diffie-Hellman (1)

$g : G$	generator of $G$
$exp(G, Z) : G$	exponentiation
$mult(Z, Z) : Z$ commutative	product in $\mathbb{Z}_q^*$
$exp(exp(z, a), b) = exp(z, mult(a, b))$	$(z^a)^b = z^{ab}$
$(g^a)^b = g^{ab}$ and $(g^b)^a = g^{ba}$ , equal by commutativity of <i>mult</i>	

$(exp(g, x) = exp(g, y)) = (x = y)$   
 $(exp'(g, x) = exp'(g, y)) = (x = y)$

**Injectivity**

**new**  $x1 : Z$ ; **new**  $x2 : Z$ ; **new**  $x3 : Z$ ; **new**  $x4 : Z$ ;  
 $mult(x1, x2) = mult(x3, x4) \approx_{1/|Z|} false$   
 $(mult(x, y) = mult(x, y')) = (y = y')$

**Collision between products**

## Other declarations for Diffie-Hellman (2)

$$\begin{aligned} & !^{i \leq N} \mathbf{new} X : G; OX() := X \\ \approx_0 \text{ [manual]} & !^{i \leq N} \mathbf{new} x : Z; OX() := \exp(g, x) \end{aligned}$$

This equivalence is very general, apply it only manually.

$$\begin{aligned} & !^{i \leq N} \mathbf{new} X : G; (OX() := X, !^{i' \leq N'} OXm(m : Z)[\text{required}] := \exp(X, m)) \\ & \approx_0 \\ & !^{i \leq N} \mathbf{new} x : Z; (OX() := \exp(g, x), !^{i' \leq N'} OXm(m : Z) := \exp(g, \text{mult}(x, m))) \end{aligned}$$

This equivalence is a particular case applied only when  $X$  is inside  $\exp$ , and good for automatic proofs.

$$\begin{aligned} & !^{i \leq N} \mathbf{new} x : Z; OX() := \exp(g, x) \\ \approx_0 & !^{i \leq N} \mathbf{new} X : G; OX() := X \end{aligned}$$

And the same for  $\exp'$ .

# Extensions for CDH

The implementation of the support for CDH required two extensions of CryptoVerif:

- An **array index  $j$  occurs as argument** of a function.
  - extend the language of equivalences used for specifying assumptions on primitives.
- The equality test  $m = \text{exp}(g, \text{mult}(b, a))$  typically occurs inside the condition of a **find**.
  - This **find** comes from the transformation of a hash function in the Random Oracle Model.

After transformation, we obtain a **find inside the condition of a find**.

# Extending the formalization of DDH in CryptoVerif

$$\begin{aligned}
 & !^{ia \leq na} \text{ new } a : Z; (OA() := \exp(g, a), Oa() := a, \\
 & \quad !^{iaDH \leq naDH} ODHa(j \leq nb) := \exp(g, \text{mult}(b[j], a))), \\
 & !^{ib \leq nb} \text{ new } b : Z; (OB() := \exp(g, b), Ob() := b, \\
 & \quad !^{ibDH \leq nbDH} ODHb(j \leq na) := \exp(g, \text{mult}(a[j], b))) \\
 & \approx \\
 & !^{ia \leq na} \text{ new } a : Z; (OA() := \exp(g, a), \\
 & \quad Oa() := \text{if } ODHa \text{ or } ODHb(ia) \text{ has been called and returned} \\
 & \quad \quad ca \text{ or } cb \text{ then event abort else } a, \\
 & \quad !^{iaDH \leq naDH} ODHa(j \leq nb) := \\
 & \quad \quad \text{if } Ob[j] \text{ or } Oa \text{ has been called then } \exp(g, \text{mult}(b[j], a)) \text{ else} \\
 & \quad \quad \text{if } cb \text{ or } ca \text{ defined for } b[j], a \text{ then that } cb \text{ or } ca \text{ else} \\
 & \quad \quad \text{new } ca : G; ca), \\
 & !^{ib \leq nb} \text{ (symmetric of } a\text{'s case)}
 \end{aligned}$$

# Extending the formalization of DDH in CryptoVerif

```

... ≈
!ia ≤ na new a : Z; (OA() := exp(g, a),
  Oa() :=
    find ua' ≤ na' st def(ka'[ua']) then event abort else
    find ub' ≤ nb', ub ≤ nb st def(kb'[ub', ub], a'[ub', ub]) ∧ a'[ub', ub] = a then
      event abort else
    let ka = mark in a,
    !ja' ≤ na' ODHa(j ≤ nb) := let b' = b[j] in
      find u ≤ nb st def(kb[u], b[u]) ∧ b' = b[u] then exp(g, mult(b', a)) else
      if def(ka) then exp(g, mult(b', a)) else
      let ka' = mark in
        find va' ≤ na' st def(b'[va'], ca[va']) ∧ b' = b'[va'] then ca[va'] else
        find vb' ≤ nb', vb ≤ nb st def(b[vb], a'[vb', vb], cb[vb', vb]) ∧ b' = b[vb] ∧
          a = a'[vb', vb] then cb[vb', vb] else
      new ca : G; ca),
!ib ≤ nb (symmetric of a's case)

```

# Extensions for DDH

The implementation of the support for DDH required two extensions of CryptoVerif:

- An **array index  $j$  occurs as argument** of a function.
  - Already done for CDH.
- Support for **abort events** in the right-hand side of equivalences.
  - Informally, when  $R$  contains abort events, the assumption  $L \approx R$  means that  $L$  is indistinguishable from  $R$  **provided the abort events are not executed!**
  - More formally, the assumption  $L \approx_p R$  implies  $C[L] \approx_{p'} C[R]$  with

$$p'(t) = \max_{C' \text{ in time } t} \Pr [C'[C[R]] \rightsquigarrow \text{abort}] + p(t + t_C)$$

When an abort event is executed, the adversary can distinguish  $C[R]$  from the  $C[L]$ .

- CryptoVerif will try to show that abort events have a negligible probability of being executed.

# Conclusion

This **formalization of Diffie-Hellman** is included in the **library of primitives** of CryptoVerif: one can use it without redefining it.

It has been used for proving protocols:

- Signed Diffie-Hellman key agreement
  - DDH + entropy extractor
  - CDH + random oracle model
- One-encrypted key exchange (OEKE, variant of EKE)
  - CDH + random oracle model + ideal cipher model

It can obviously still prove

- El Gamal (DDH)
- Hashed El Gamal
  - DDH + entropy extractor
  - CDH + random oracle model