Prime, Order Please!
Revisiting Small Subgroup and Invalid Curve Attacks on Protocols using Diffie-Hellman

Dennis Jackson

Cas Cremers

University of Oxford

EPSRC Centre for Doctoral Training in Cyber Security

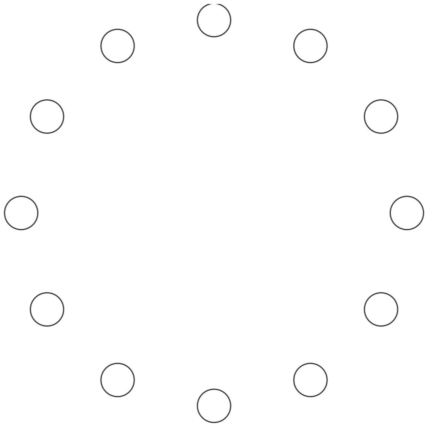
CISPA Helmholtz Center for Information Security

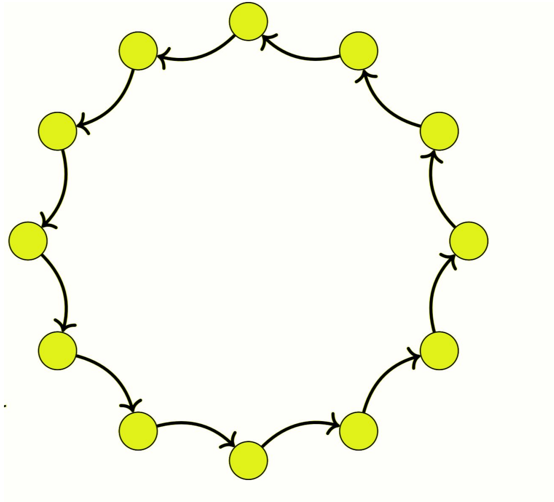
Assumptions about DH Groups

Let G be a finite cyclic group of prime order p with $p = O(2^k)$ for some security parameter k and let g be a generator of the group G. Further, let $KDF : \{0,1\}^* \rightarrow \{0,1\}^k$ denote a key derivation function.

Prime Order Groups

- One special element, the identity: $\mathtt{id}^x = \mathtt{id}$
- Every other element is a generator
- A generator raised to a power can become any other element
- Uniform structure

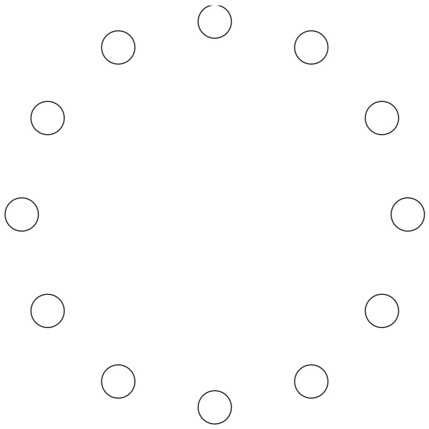


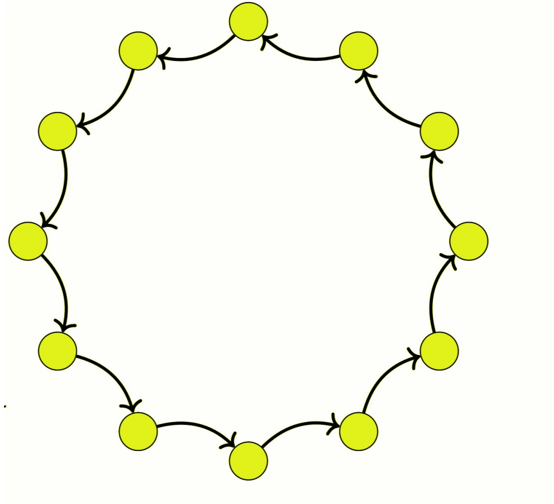


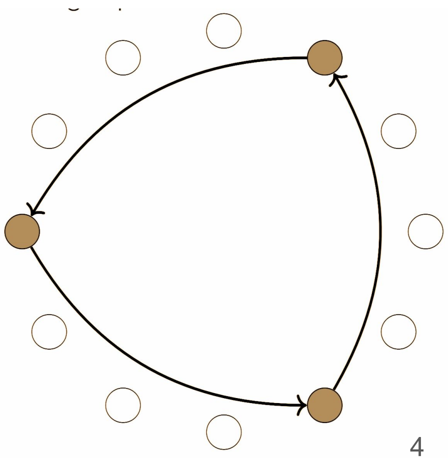
Non-Prime Order Groups

Non-prime order groups have a more intricate structure, which leads to additional behaviour.

Some elements have small order and become trapped in small subgroups.







Consequences of Non-Prime Order Groups: I

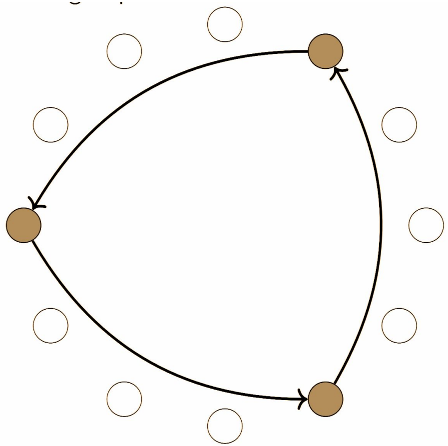
R knows $\mathtt{r}$

R receives: $\mathtt{g^{i}, senc(g^{ir}, M)}$

R computes $\mathtt{g^{ir}}$ decrypts the ciphertext and learns $\mathtt{M}$

R claims: $\mathtt{g^{i}}$ knows $\mathtt{g^{r}}$ and sent $\mathtt{M}$ to me

Consequences of Non-Prime Order Groups: I



R knows $\mathtt{r}$

R receives: $\mathtt{g^{i},senc(g^{ir},M)}$

R computes $\mathtt{g^{ir}}$ decrypts the ciphertext and learns $\mathtt{M}$

R claims: $\mathtt{g^{i}}$ knows $\mathbf{g^{r}}$ and sent $\mathtt{M}$ to me

What if **X** sends **h**, a small subgroup element? **Non-contributory behaviour**

$\mathbf{h}^{r}$ can only taken one of a small number of possibilities. **X** can guess the outcome and send the associated ciphertext, despite not knowing $\mathbf{g}^{r}$

Consequences of Non-Prime Order Groups II

R accepts a public key $\mathbf{g^{i}}$ and calculates the shared secret $\mathbf{g^{ir}}$

R never accepts the same public key twice.

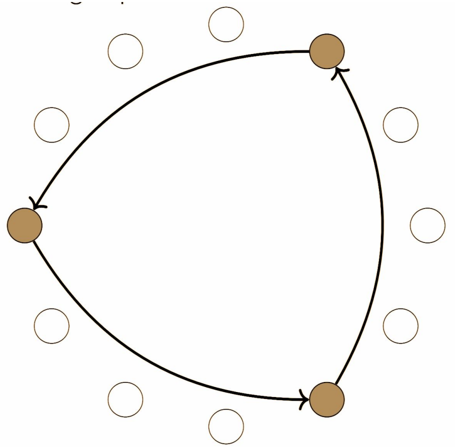
R claims the resulting shared secrets are unique

Consequences of Non-Prime Order Groups II

R accepts a public key $\mathbf{g^i}$ and calculates the shared secret $\mathbf{g^{ir}}$

R never accepts the same public key twice.

R claims the resulting shared secrets are unique



X sends their public key $\mathbf{g^x}$. Then sends $\mathbf{hg^x}$.

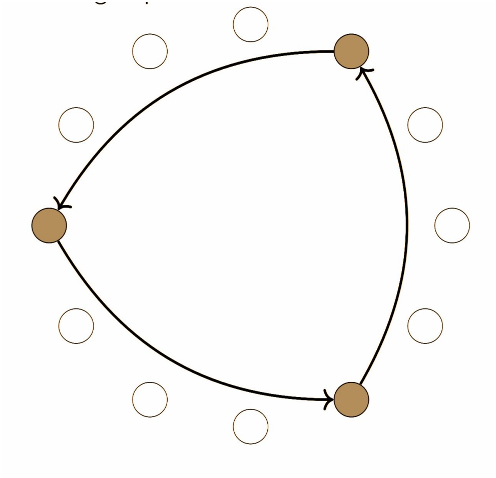
$\mathbf{hg^x \;!=\; g^x}$ yet $\mathbf{(hg^x)^r = h^r g^{xr} = g^{xr}}$ with non-negligible probability.

Distinct elements can result in **equivalent** outcomes

Consequences of Non-Prime Order Groups III

Let $\mathbf{x}$ is a secret exponent and $\mathbf{h_i}$ is a sequence of distinct small subgroup generators

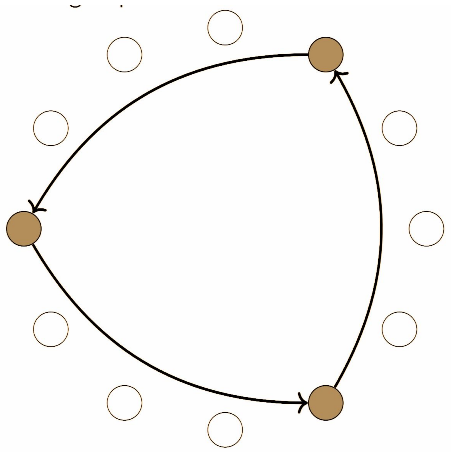
Under exponentiation: $\mathbf{h_i}^{\mathbf{x}} = \mathbf{h_i}^{(\mathtt{x\ mod\ |h|})}$



Consequences of Non-Prime Order Groups III

Let $\mathbf{x}$ is a secret exponent and $\mathbf{h_i}$ is a sequence of distinct small subgroup generators

Under exponentiation: $\mathbf{h_i^x = h_i^{(x \bmod |h|)}}$



With $|\mathbf{h_i}|$ small, the attacker can calculate $\mathbf{x \bmod |h_i|}$

Repeat multiple times, combine with Chinese Remainder Theorem to learn $\mathbf{x}$ through **key leakage**

Popularity of Non-Prime Order Groups

- Finite Fields cannot be prime order
- The fastest elliptic curves are not prime order
  - Curve25519
  - Curve448
  - Curve4Q

- Mitigations can be employed at the protocol level to prevent unwanted behaviour

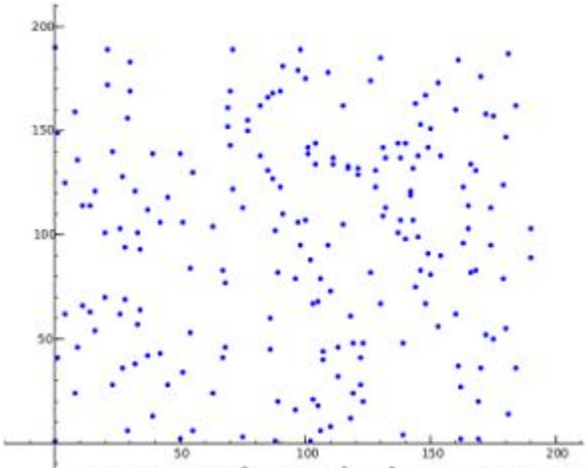
Further Assumptions about DH Groups

groups, where group elements are represented as points in a finite plane. All ECC cryptosystems implicitly assume that only valid group elements will be processed by the different cryptographic algorithms. It is well-known that a check for group membership of given points in the plane should be performed before processing.

Invalid Elements

Points on an Elliptic Curve are two finite field elements which satisfy the curve equation.

NIST P224 has ~2^{224} elements in a space of ~2^{448} points.

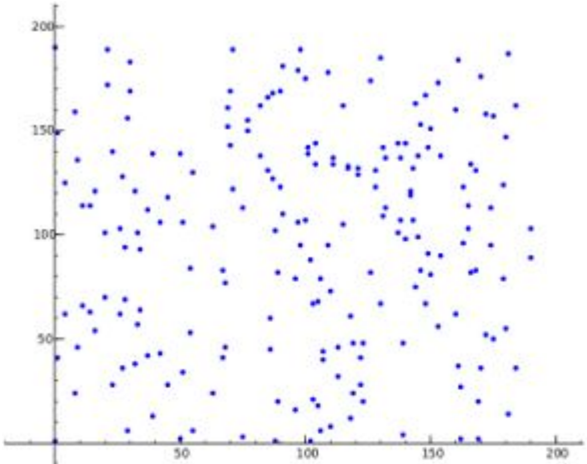


Elliptic Curve $y^2 + xy = x^3 + x^2 + 1$ over $GF(191)$

Consequences of Invalid Elements

Operating on invalid points can force you onto a different curve which might have:

- A small subgroup (leading to non-contributive behaviour or equivalent points)
- Many small subgroups (leading to key leakage)
- Easy discrete logarithms



Elliptic Curve $y^2 + xy = x^3 + x^2 + 1$ over $GF(191)$

Assumptions about DH Groups

Let G be a finite cyclic group of prime order p with $p = O(2^k)$ for some security parameter k and let g be a generator of the group G. Further, let $KDF : \{0,1\}^* \rightarrow \{0,1\}^k$ denote a key derivation function.

groups, where group elements are represented as points in a finite plane. All ECC cryptosystems implicitly assume that only valid group elements will be processed by the different cryptographic algorithms. It is well-known that a check for group membership of given points in the plane should be performed before processing.

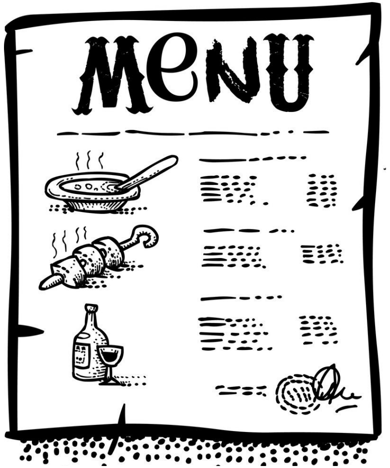
Recent Attacks

Year	Violated Assumption	Affected	Impact	Ref
2015	Valid Elements	Java's Default TLS Library Bouncy Castle	Server Key Recovery	[1]
2016	Prime Order Group	OpenSSL Exim mail server Unbound DNS client	Server Key Recovery	[2]
2017	Valid Elements	JSON Web Encryption	Server Key Recovery	[3]
2018	Valid Elements	Bluetooth Secure Pairing	Session Key Recovery	[4]

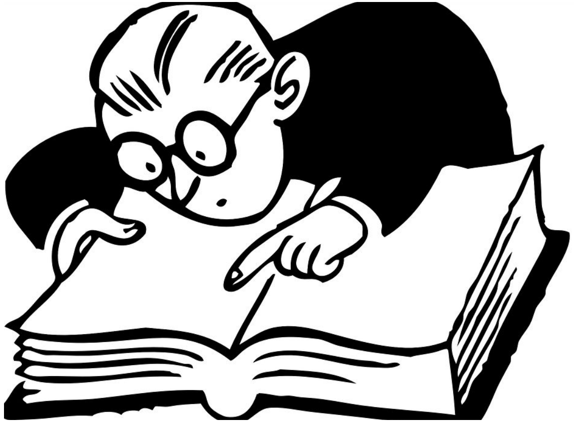
[1] Jager, Tibor, Jörg Schwenk, and Juraj Somorovsky [2] Valenta, Luke, et al [3] Quan Nguyen and Antonio Sanso [4] Biham, Eli, and Lior Neumann

Why does this keep going wrong?

Performance



Choice



Fine Print

Tamarin's Symbolic DH Model

Model

```
g/0 ^/2 */2 (-1)/1 (1)/0
```

$(x^{y})^{z} = x^{y*z}$
$x^{1} = x$

$x * (y * z) = (x * y) * z$
$x * y = y * x$
$x * 1 = x$
$x * x^{-1} = 1$
$(x^{-1})^{-1} = x$

Tamarin's Symbolic DH Model

Model

`g/0 ^/2 */2 (-1)/1 (1)/0`

$(x^y)^z = x^{y*z}$
$x^1 = x$

$x * (y * z) = (x * y) * z$
$x * y = y * x$
$x * 1 = x$
$x * x^{-1} = 1$
$(x^{-1})^{-1} = x$

Implicit Assumptions

- Reject the identity element
- Prime Order Group
- Only operate on valid elements

Extension: the Identity Element

Extended Model

```
g/0 ^/2 */2 (-1)/1 (1)/0
id/0
```

$(x^y)^z = x^{y*z}$
$x^1 = x$
$id^x = id$

$x * (y * z) = (x * y) * z$
$x * y = y * x$
$x * 1 = x$
$x * x^{-1} = 1$
$(x^{-1})^{-1} = x$

Implicit Assumptions

- ~~Reject the identity element~~
- Prime Order Group
- Only operate on valid elements

Note: This equation is similar to a ProVerif model used by Bhargavan, Delignat-Lavaud, and Pironti in 2015.

Extension: Non-Prime Groups

Let $\mathbf{G}$ be some cryptographically relevant group of non-prime order:

$$\mathbf{G} \cong \mathbf{H} \times \mathbf{Z}_p$$

$$g^x \cong (s^x, n^x)$$

Extension: Non-Prime Groups

Let $\mathbf{G}$ be some cryptographically relevant group of non-prime order:

$$G \cong H \times Z_p$$

$$g^x \cong (s^x, n^x)$$

(id, g^x) (h^y, id) (id, id) (h^y, g^x)

Extension: Non-Prime Groups

Let $\mathbf{G}$ be some cryptographically relevant group of non-prime order:

$$G \cong H \times Z_p$$

$$g^x \cong (s^x, n^x)$$

(id, g^x) (h^y, id) (id, id) (h^y, g^x)

NIST P224	$\cong Z_p$	DSA Group	$\cong \ldots \times Z_p$
Curve25519	$\cong Z_8 \times Z_p$	Safe Prime Group	$\cong Z_2 \times Z_p$

Modelling the non-prime group

1. Model elements as a pair **`e(s,n)`** ∈ **`H x G`**

Modelling the non-prime group

1. Model elements as a pair **`e(s,n)`** ∈ **H x G**

2. Model exponentiation in **H** as an attacker controlled oracle

```
[In(X),State(y)]--[]->[Out(X^y)]

[In(e(s,n)),State(y),In(a)]--[Raised(s,a,y)]->[Out(e(a,n^y))]
```

Modelling the non-prime group

1. Model elements as a pair **`e(s,n)`** $\in$ **`H x G`**

2. Model exponentiation in **H** as an attacker controlled oracle

`[In(X),State(y)]--[]->[Out(X^y)]`

`[In(e(s,n)),State(y),In(a)]--[Raised(s,a,y)]->[Out(e(a,n^y))`

3. Impose minimal restrictions on the oracle

Consistency: $\texttt{Raised(s, }a_1\texttt{, y)} \wedge \texttt{Raised(s, }a_2\texttt{, y)} \Rightarrow a_1 = a_2$

Identity: `Raised(id, a, y)` $\Rightarrow$ `a = id`

Example - Traditional Model

R receives: $\mathbf{g^{i}}$, $\mathbf{senc(g^{ir},M)}$

R computes $\mathbf{g^{ir}}$ decrypts the ciphertext and learns **M**

R claims: $\mathbf{g^{i}}$ knows $\mathbf{g^{r}}$ and sent **M** to me

[In(g^{i},ct),State(r)]

--[]->

[Assert(g^{i},adec(ct,g^{ir})]

Example - Transformed Model

R receives: $\mathtt{g^{i},senc(g^{ir},M)}$

R computes $\mathtt{g^{ir}}$ decrypts the ciphertext and learns $\mathtt{M}$

R claims: $\mathtt{g^{i}}$ knows $\mathtt{g^{r}}$ and sent $\mathtt{M}$ to me

```
[In(e(s,n),ct),State(r),In(a)]
--[Raised(s,a,r)]->
[Assert(e(s,n),adec(ct,e(a,nʳ))
```

Example - Non Contributory Behaviour

R receives: $\mathtt{g^{i}, senc(g^{ir}, M)}$

R computes $\mathtt{g^{ir}}$ decrypts the ciphertext and learns $\mathtt{M}$

R claims: $\mathtt{g^{i}}$ knows $\mathtt{g^{r}}$ and sent $\mathtt{M}$ to me

$\mathtt{[In(e(h_1,id),ct),State(r),In(h_2)]}$

$\mathtt{--[Raised(h_1,h_2,r)]->}$

$\mathtt{[Assert(e(h_1,id),adec(ct,e(h_2,id))}$

In the Paper

- Detecting Key Leakage
- Invalid EC Elements
- Mitigations
- Parameterised Models

Case Study: Secure Scuttlebutt

“sea-slang for gossip”

a decentralised secure gossip platform

Whitelisted by Mozilla for Firefox support

Peers have their own long term public identity key

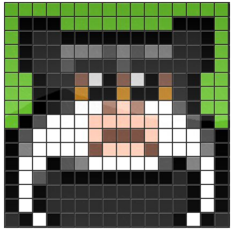
‘Friend’ each other by exchanging public keys

Scuttlebutt Security Properties

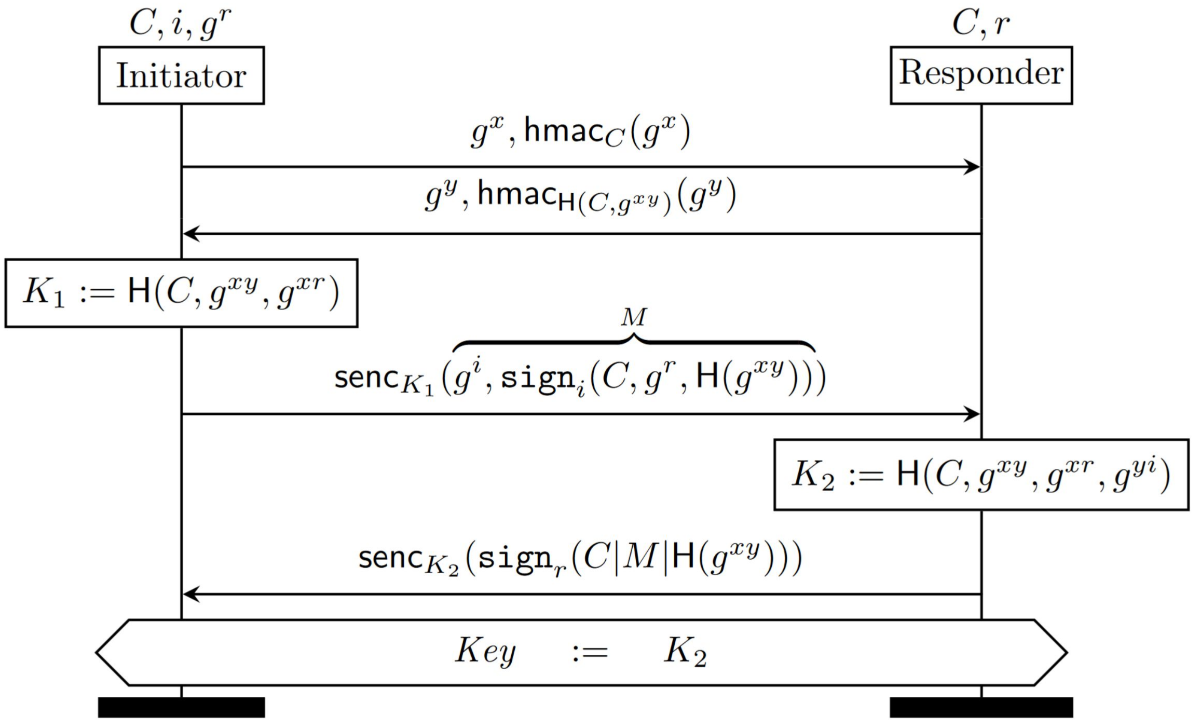
- Peers create channels using a bespoke key exchange
- Ensures authentication, forward secrecy, identity hiding.
- Previous Tamarin Verification
- Uses Curve25519 which is not prime order

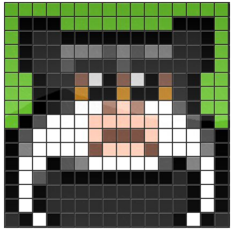
Key Authentication Requirement:

Initiator must prove knowledge of Responder’s **public** key

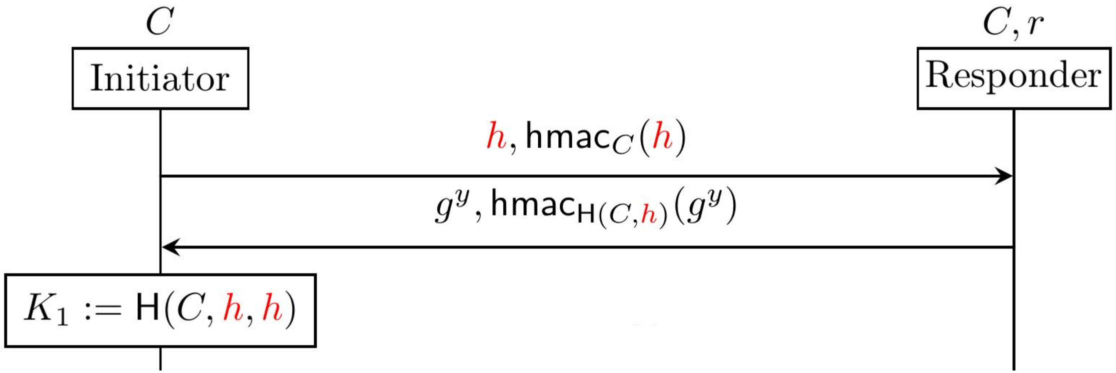


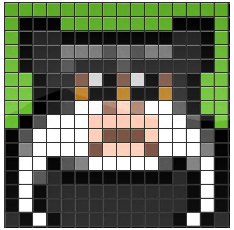
Scuttlebutt: Key Exchange



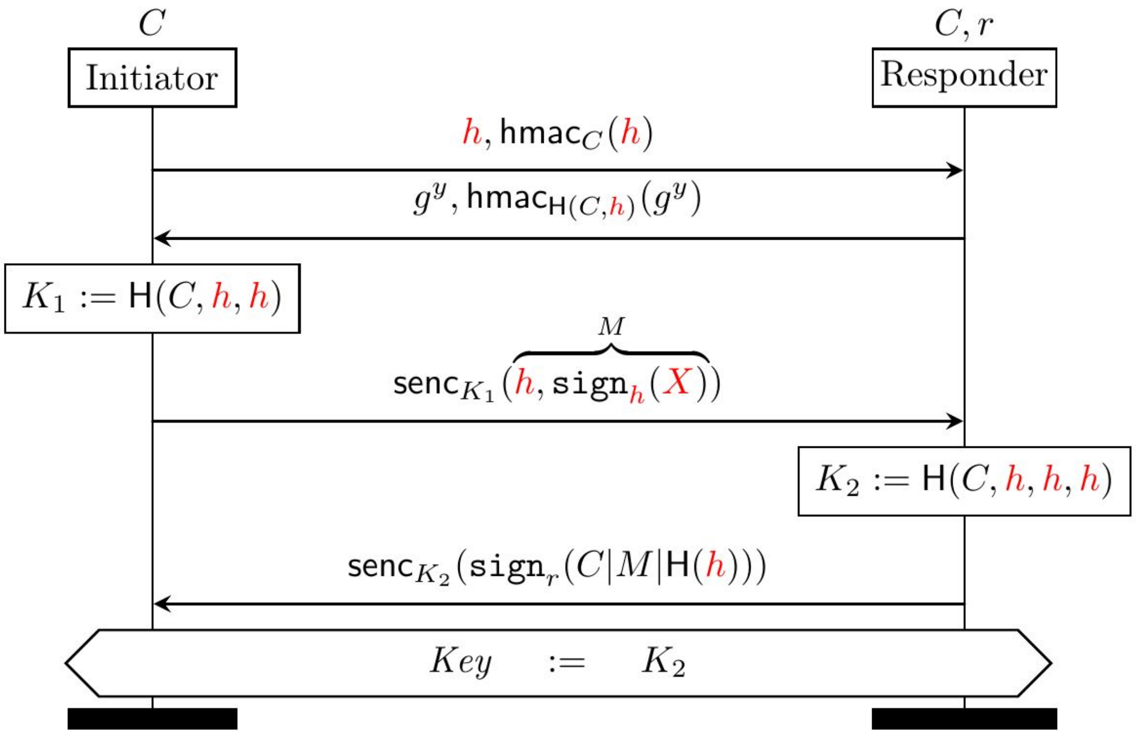


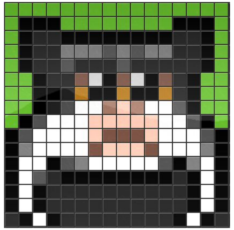
Scuttlebutt Attack (½)





Scuttlebutt Attack 2/2





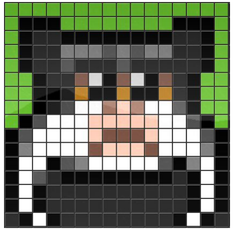
Scuttlebutt Fix

There are two natural fixes:

- Reject low order points
- Add raw identities to KDF, in addition to shared key

Scuttlebutt opted to reject low order points as it is backwards compatible. However, such checks could be silently omitted by a faulty implementation.

Protocol	**Variant**	**Secure?**	**T (min)**
Scuttlebutt Curve25519	Original	●	131
	With exclusion of low order points	✓	88
	Including identities in the KDF	✓	<1



Scuttlebutt Fix

There are two natural fixes:

- Reject low order points
- Add raw identities to KDF, in addition to shared key

Scuttlebutt opted to reject low order points as it is backwards compatible. However, such checks could be silently omitted by a faulty implementation.

Protocol	Variant	Secure?	T (min)
Scuttlebutt Curve25519	[illegible]	●	131
	[illegible]	✓	88
	[illegible]	✓	<1

100% Automated
No Added Heuristics
or Lemmas

Do NaCl libraries reject low order points?

LibSodium

x/crypto/nacl
Golang

HACL
Project Everest

CIRCL
Cloudflare

Do NaCl libraries reject low order points?

	LibSodium	x/crypto/nacl Golang	HACL Project Everest	CIRCL Cloudflare
Before	**Secure**	**Insecure**	**Insecure**	**Insecure**

Do NaCl libraries reject low order points?

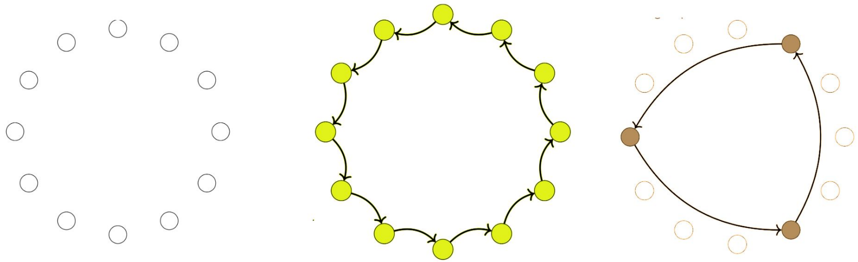
	LibSodium	x/crypto/nacl Golang	HACL Project Everest	CIRCL Cloudflare
Before	**Secure**	**Insecure**	**Insecure**	**Insecure**
Now	**Secure**	**Patched in 1.13**	**Intent to Patch**	**Patched**

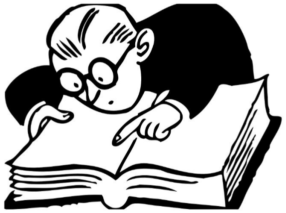
Future Work

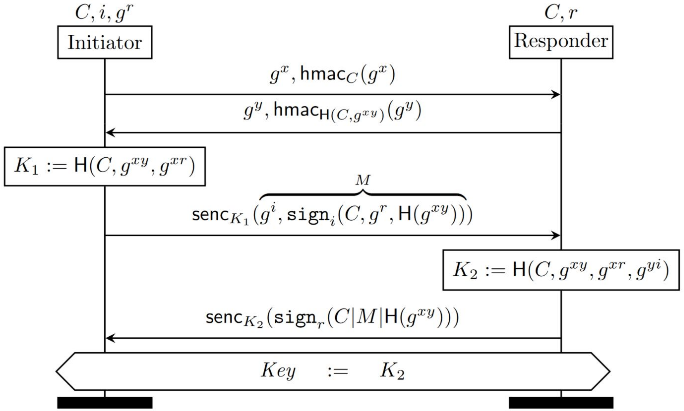
- Although we know we find strictly more attacks than traditional DH models, we have no proof of **computational soundness.**

- With symbolic models becoming ever more granular, **automatically generating models** from reference implementations looks increasingly attractive

Dennis Jackson, Cas Cremers @ CSF 2019



		
Performance	Choice	Fine Print



	 LibSodium	 x/crypto/nacl Golang	 HACL Project Everest	CIRCL Cloudflare
Before	**Secure**	**Insecure**	**Insecure**	**Insecure**
Now	**Secure**	**Patched in 1.13**	**Intent to Patch**	**Patched**

39

Thanks for Listening!

dennis.jackson@cs.ox.ac.uk

Fine Print: How do I validate Curve25519 public keys?

Don't. The Curve25519 function was carefully designed to allow all 32-byte strings as Diffie-Hellman public keys. [...]

There are some **unusual non-Diffie-Hellman elliptic-curve protocols** that need to ensure ‘**contributory**’ behavior. In those protocols, you should reject the 32-byte strings that, in little-endian form, represent 0, 1, [...]. **But these exclusions are unnecessary for Diffie-Hellman.**

- Daniel Bernstein, designer of Curve25519

Key Theorem

Theorem 3 (The fundamental theorem of finite abelian groups). *Let G be a finite abelian group of order n. Let the unique factorisation of n into distinct prime powers be given by $n = p_1^{a_1} \ldots p_k^{a_k}$. Then:*

1. $G \cong A_1 \times \ldots \times A_k$ *where* $|A_i| = {p_i}^{a_i}$
2. *For each* $A \in \{A_1, \ldots, A_k\}$ *with* $|A| = p^a$

$$A \cong \mathbb{Z}_{p^{b_1}} \times \ldots \times \mathbb{Z}_{p^{b_t}}$$

   with $b_1 \geq b_2 \geq \ldots \geq b_t$ *and* $b_1 + \ldots + b_t = a$

3. *The decomposition given above is unique.*

Modeling Key Leakage Attacks

Preconditions for a successful attack

a. The same (secret) exponent must be used in multiple calculations
b. The protocol must operate on an element with a low order component
c. The attacker must be able to learn or guess the result
d. The group order must have enough small factors to allow for a recovery of a significant portion of the key.

a, b, d can be described as a trace property.

c is a type of strong secrecy.

Elliptic Curve Elements

We split our elements into a 2-tuple (again):

$$\mathbf{g = (x,y) = ((x_s,x_n),(y_s,y_n))}$$

We say $(\mathbf{x},\mathbf{y})$ is a valid point when $\mathbf{x} = \mathbf{y}$. Justified as $\mathbf{x}$ defines $\mathbf{y}$ up to sign.

If the protocol performs an exponentiation on an invalid point, we let the attacker choose the outcome.

We also provide a capability for the attacker to take discrete logs on invalid points.

Modelling Mitigations

- Rejecting the identity element
- Rejecting low order points
- Checking the order of an element
- Clearing low order bits (raising by the cofactor)
- Checking the curve equation
- Using a single coordinate ladder

Each implemented with a specific action and restriction.

Models Summary

- Internal Group Structure
  - Prime Order
  - Non Prime Order
- Group Elements
  - Finite Fields
  - Elliptic Curve Elements
- Mitigations

Also supports multiple groups of different types in the same protocol.