# Projective Coordinates Leak

Marc Stoecklin

SSC

Semester Project

**Responsible**

Prof. Serge Vaudenay
EPFL / LASEC

**Supervisors**

Jean Monnerat
Thomas Baignères
EPFL / LASEC

February 16, 2006

# Overview

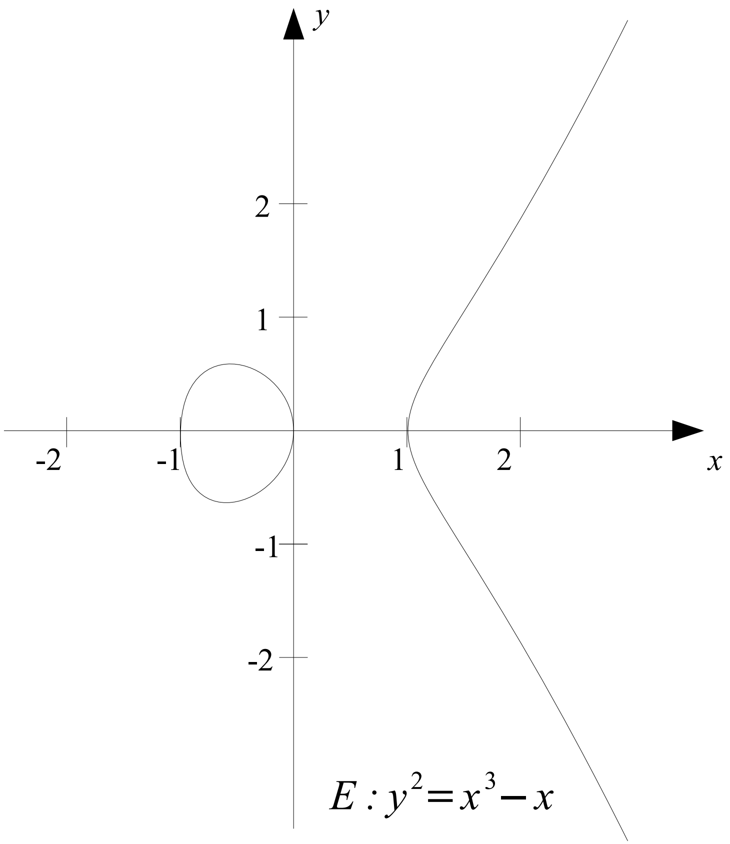
1. Elliptic Curves and Projective Coordinates
2. Description of the Attack
3. Required Operations on Finite Fields
4. Implementation Issues
5. Results and Analysis
6. Thwarting the Attack
7. Conclusion

# Elliptic curves

- **Defined by a polynomial equation**
  - over a field, e.g., the finite field $\boldsymbol{F_p}$
- $\boldsymbol{F_p}$ **with** $p \neq 2, 3$

$$E : y^2 = x^3 + ax + b$$

where $a, b \in \boldsymbol{F_p}$.




- **Affine coordinates**
  - unique representation of an element

$$P = (x, y)$$

# Operations on Affine Coordinates

- **Problem**
  - Affine arithmetic formulae require field inversions (divisions)
  - Computation is expensive
- **Solution**
  - Represent points in projective coordinates
  - Perform computation with division-free formulae
- **Advantage**
  - Single field inversion required to convert back to affine representation

# Projective Coordinates

- **Representation of curve points**

**Affine representation** **Projective Representation**

$$P=(x,y) \qquad P=(X:Y:Z)$$

- **Jacobian Projective coordinates**

$$(x,y)=\left(\frac{X}{Z^{2,}}\frac{Y}{Z^{3}}\right) \qquad (X:Y:Z)=\left(\lambda^{2}x:\lambda^{3}y:\lambda\right)$$

  – There exist $p - 1$ projective representations of a curve point

# Attack on Projective Coordinates

- Proposed by Naccache, Smart, and Stern at Eurocrypt 2004
- **Setup**
  - Elliptic curve $E$ with public base point $G$ of prime order
- **Application**
  - Secret exponent $k$
  - Projective double-and-add multiplication to obtain $P = [k]G$
  - e.g., Signatures/Diffie-Hellman Key Exchange
- **Question**
  - *Does the projective representation of $P$ reveal information about $k$?*

# Double-and-add multiplication

## Binary left-to-right multiplication

INPUT: $k = (k_{t-1}, \dots , k_1 , k_0), P \in E$

OUTPUT: $Q = [k]P \in E$

$Q \leftarrow O$
**for** $i = t - 1$ **downto** 0 **do**
    $Q \leftarrow [2]Q$
    **if** $k_i = 1$ **then**
        $Q \leftarrow Q + P$
    **end if**
**end for**
Return($Q$)

# Attack scheme

- **Computation of** $P = [k]G$

$$G \Rightarrow M_L \Rightarrow \dots \Rightarrow M_2 \Rightarrow M_1 \Rightarrow P$$

- **Attack procedure**
  - The last operation in the double-and-add multiplication to obtain $P$ from the intermediate point $M_1$ was either

$$P = [2]M_1 \qquad \text{or} \qquad P = M_1 + G$$

  - The adversary obtains the projective representation of $P$
  - He guesses on the last bit and inverts the last operation

# Attack scheme: Inverting a doubling

- **Assumption**: $M_i \Rightarrow M_{i\text{-}1}$ is a doubling
  - find affine intermediate point $M_i$ (unique point)

$$M_i = (x_i, y_i) = [2^{-1}] M_{i-1}$$

  - consider the projective Jacobian doubling formula

$$Z_{i-1} = 2 Y_i Z_i = 2 y_i Z_i^4 \qquad \Rightarrow \qquad Z_i = \sqrt[4]{\frac{Z_{i-1}}{2 y_i}}$$

  - extract fourth root in $\boldsymbol{F_p}$ to find projective representations of $M_i$:
    - if $p \equiv 1 \pmod 4$: possible in half of the cases, yields 2 solutions
    - if $p \equiv 3 \pmod 4$: possible for one quarter, yields 4 solutions

# Attack scheme: Inverting an addition

- **Assumption**: $M_i \Rightarrow M_{i\text{-}1}$ is an addition
  - find affine intermediate point $M_i$ (unique point)

$$M_i = (x_i, y_i) = M_{i-1} - G$$

  - consider the projective Jacobian addition formula

$$Z_{i-1} = (x_i - x_G) Z_i^3 \qquad \Rightarrow \qquad Z_i = \sqrt[3]{\frac{Z_{i-1}}{x_i - x_G}}$$

  - extract cube root in $\boldsymbol{F_p}$ to find projective representation(s) of $M_i$:
    - if $p \equiv 1 \pmod 3$: possible a third of all cases, yields 3 solutions
    - if $p \equiv 2 \pmod 3$: always possible, yields 1 solution

# Attack scheme: Tree

- assume bit $k_i = 1$
- subtract $G$ from $M_j$
- extract cube roots
- iterate on the solutions by halving

- assume bit $k_i = 0$
- halve the affine point $M_j$
- extract fourth roots to find the set of admissible projective representations

$M_j, Z_j$

$M_{j-1}^{(1)} = M_j - G$

$Z_{j-1}^{(1)}$

$M_{j-2}^{(1)} = [2^{-1}] M_{j-1}^{(1)}$

$Z_{j-2}^{(1)}$

$M_{j-1}^{(0)} = [2^{-1}] M_j$

$Z_{j-1}^{(0)}$

# Attack analysis

- For Jacobian projective coordinates, reversing a bit may lead to a set of pre-images depending on $p$:

| p mod 12 | $P \to P{+}G$ | $P \to [2]P$ | $k_i = 0$ | $k_i = 1$ |
|---|---|---|---|---|
| 1 | $3 \hookrightarrow 1$ | $4 \hookrightarrow 1$ | 4 pre-images | 12 pre-images |
| 5 | $1 \hookrightarrow 1$ | $4 \hookrightarrow 1$ | 4 pre-images | 4 pre-images |
| 7 | $3 \hookrightarrow 1$ | $2 \hookrightarrow 1$ | 2 pre-images | 6 pre-images |
| 11 | $1 \hookrightarrow 1$ | $2 \hookrightarrow 1$ | 2 pre-images | 2 pre-images |

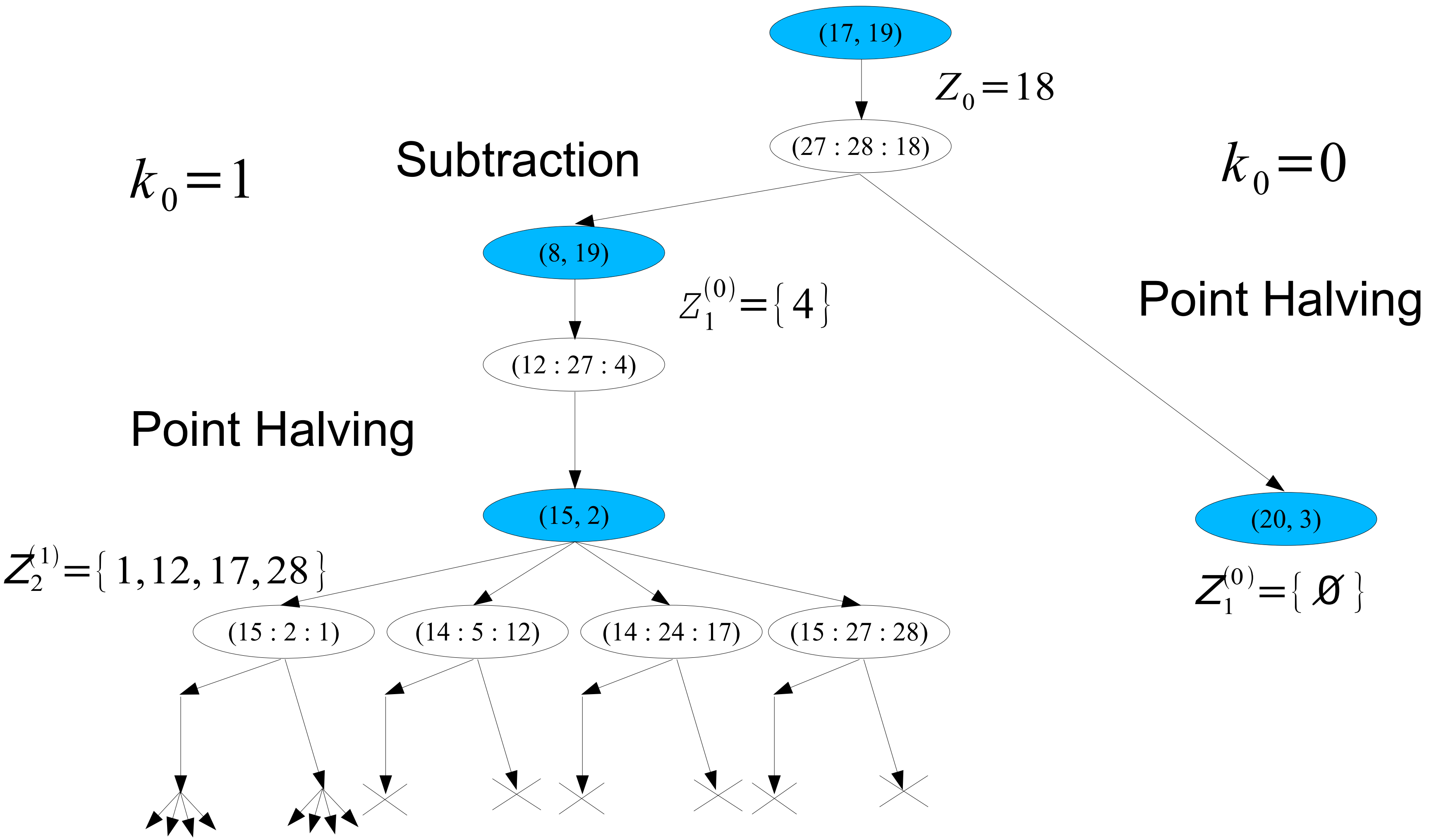
- Pr["$(M_i, Z_i)$ has pre-images"] = $\dfrac{1}{\text{\# of pre-images}}$

# Illustrative Example (1)

- **Setup**
  - Curve $E: y^2 = x^3 + 4x + 20$ defined over $\boldsymbol{F}_{29}$
  - Base Point $G = (5, 22)$
- **Multiplication**
  - Secret $k = 11_{10} = 1011_2$
  - Result $P = [k]G = (27 : 28 : 18)$

| **Assumption**: $k_0 = 0$ | **Assumption**: $k_0 = 1$ |
| --- | --- |
| solve $Z = \sqrt[4]{3} \pmod{29}$ $\Rightarrow$ no solutions: $k_0 \neq 0$ | solve $Z = \sqrt[3]{6} \pmod{29}$ $\Rightarrow$ 1 solution |

# Illustrative Example (2)

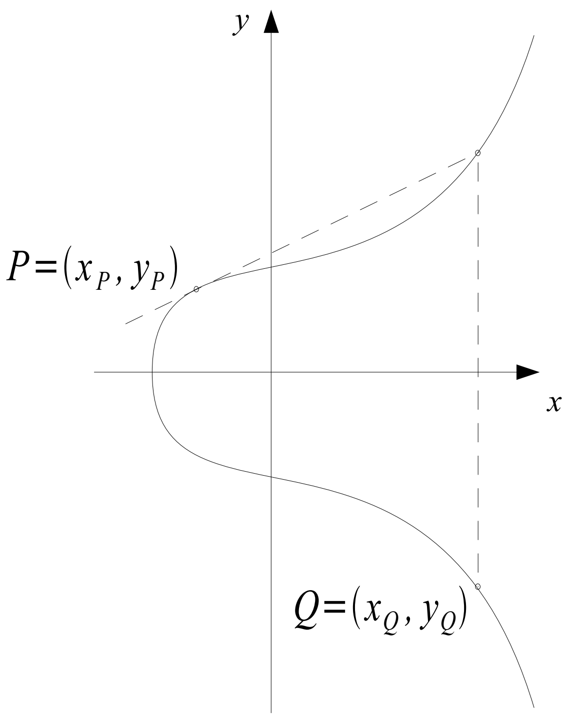


# Required Operations for the Attack

- Find cube roots in $\boldsymbol{F}_p$
- Find fourth roots in $\boldsymbol{F}_p$
  - Reduction to two consecutive square roots in $\boldsymbol{F}_p$
- Point Halving on a curve $E$
  - There exists no closed-form formula as for addition/doubling
  - Point doubling is not necessarily injective
- Factorizing polynomials in $\boldsymbol{F}_p$

# Point Halving

- **Formulae for point halving**
  - by knowing the order of the group (too limiting)
  - by deriving geometrically from the curve equation (factorizations/expensive)
  - by reversing the formulae for doubling (factorization of a polynomial of deg. 4)




- **Factorize the polynomial**

$$F(x_P)=x_P^4-4x_Q x_P^3-2a\,x_P^2+(8x_Q^3+4x_Q a-8y_Q^2)x_P+4x_Q^4+4x_Q^2 a+a^2-4x_Q y_Q^2$$

- **Cantor-Zassenhaus algorithm**

# Implementation with GMP

- **GMP**
  - GNU Multiple Precision Arithmetic Library from “Swox AB”, Sweden
- **Implementation**
  - Elliptic curve and finite field arithmetic
  - Attack by knowing the order of the curve
  - Factorization with Cantor-Zassenhaus provided by NTL
- **Issues**
  - Problematic converting: GMP to NTL representation and vice versa
  - Efficiency, e.g., finding a generator in $\boldsymbol{F}_p$ or counting curve points

# Implementation with LiDIA

- **LiDIA**
  - C++ library written by “Institute for Computer Algebra, Distributed Systems and Cryptography”, TU Darmstadt
  - Built-in finite field and elliptic curve arithmetics
  - Cantor-Zassenhaus algorithm for factorization
- **Issues**
  - Compilation problems with recent compilers (Latest stable: Dec 04)
  - Bugs in the code (bug report filed)

# Attack Library

- **Wrapper class** `ECC_wrapper`
  - double-and-add multiplication
  - point halving (by using the Cantor-Zassenhaus algorithm)
- **Utilities class** `FF_utils`
  - cube and fourth roots algorithm
- **Attack class** `Attack`
  - launch an attack on a projective point
  - obtain statistics (Graphviz, AWK, Gnuplot)

# Results and Analysis

- **Visualization of a sample attack run**
- **Determining a few trailing bits of the secret**
  - random curves and random base points
  - curves defined by the NIST
- **Key ranking and position expectation**
  - random curves and random base points
  - curves defined by the NIST

# Visualization of an Attack Run (1) DEMO[1]

## Configuration

- Curve $E: y^2 = x^3 - 3x + 12$ defined over $\boldsymbol{F}_{12301}$ (16 bits)
- Base Point $G = (3111, 10607)$
- Secret $k = 2120_{10} = 100001001000_2$

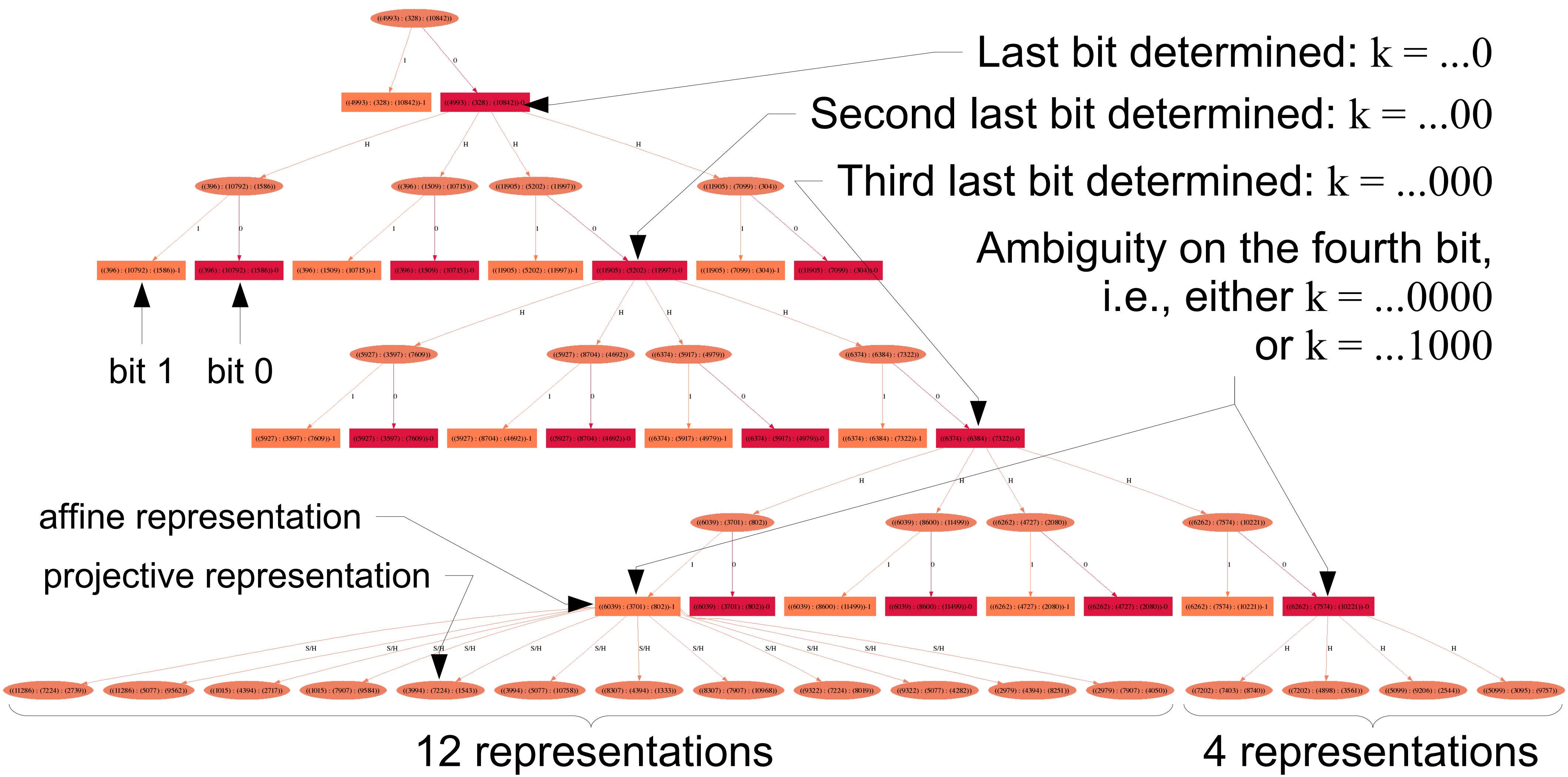
## Recall

| p mod 12 | $P \rightarrow P+G$ | $P \rightarrow [2]P$ | $k_i = 0$ | $k_i = 1$ |
|---|---|---|---|---|
| 1 | $3 \hookrightarrow 1$ | $4 \hookrightarrow 1$ | 4 pre-images | 12 pre-images |

## Attack run

- attempt to determine the last 4 bits

# Visualization of an Attack Run (2) DEMO$_1$



# Determining trailing bits (1)

DEMO$_2$

| p mod 12 | 1 | 5 | 7 | 11 |
|---|---|---|---|---|
| Pr[“parity even \| k even”] | 0.938 | 0.773 | 0.844 | 0.512 |
| Pr[“parity odd \| k odd”] | 0.873 | 0.768 | 0.504 | 0.469 |
| Pr[“bit 0 correct”] | 0.895 | 0.771 | 0.674 | 0.490 |
| Pr[“bit 1 correct”] | 0.518 | 0.107 | 0.187 | 0.000 |
| Pr[“bit 2 correct”] | 0.200 | 0.000 | 0.059 | 0.000 |
| Pr[“bit 3 correct”] | 0.041 | 0.000 | 0.022 | 0.000 |
| Pr[“bit 4 correct”] | 0.010 | 0.000 | 0.005 | 0.000 |
| Pr[“bit 5 correct”] | 0.003 | 0.000 | 0.001 | 0.000 |

1024 experiments on random curves over a field of 32 bits

# Determining trailing bits (2)

| NIST curve<br>p mod 12 | P-192<br>11 | P-224<br>1 | P-256<br>7 | P-384<br>11 | P-512<br>7 |
|---|---|---|---|---|---|
| Pr[“parity even \| k even”] | 0.521 | 1.000 | 0.822 | 0.480 | 0.861 |
| Pr[“parity odd \| k odd”] | 0.498 | 1.000 | 0.484 | 0.492 | 0.482 |
| Pr[“bit 0 correct”] | 0.510 | 1.000 | 0.653 | 0.486 | 0.672 |
| Pr[“bit 1 correct”] | 0.000 | 0.996 | 0.139 | 0.000 | 0.140 |
| Pr[“bit 2 correct”] | 0.000 | 0.980 | 0.040 | 0.000 | 0.042 |
| Pr[“bit 3 correct”] | 0.000 | 0.953 | 0.019 | 0.000 | 0.017 |
| Pr[“bit 4 correct”] | 0.000 | 0.953 | 0.005 | 0.000 | 0.006 |
| Pr[“bit 5 correct”] | 0.000 | 0.594 | 0.002 | 0.000 | 0.001 |

1024 experiments on NIST curves (256 runs for P-224)

# True Sequence Position Expectation (TSPE)

- **Question**
  - *Is it possible to obtain a meaningful statistic on trailing bit sequences of length $\ell$ of the secret $k$?*
- **Key ranking**
  - Attack on $\ell$ bits: count the representations per admissible sequence
  - Determine the position of the $\ell$ last bits of $k$ in the key ranking: **Expected position of the true sequence** $pos_{true,k}$
- **Average expected position**
  - Repeat the key ranking for distinct values of $k$ for a configuration
  - **Average expected position** $\mu_{pos} = \sum_k pos_{true,k}$

# Establishing the Key Ranking

DEMO3

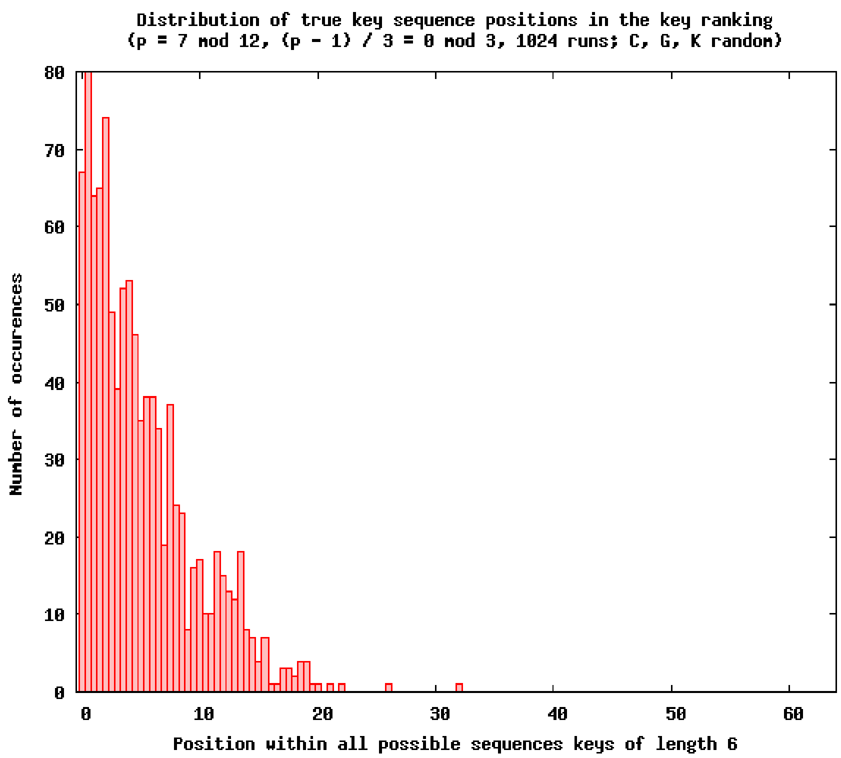
- **Example**
  - $\ell = 4$
  - 16 possible bit sequences
  - $k = 2750_{10} = 10101011\mathbf{1110}_2$
  - Expected position:

$$pos_{true} = \frac{3-1}{2} = 1$$

| Index | Bit sequence | Possible representations |
|---|---|---|
| 0 | 0110 | 36 |
| 1 | 1100 | 36 |
| **2** | **1110** | **36** |
| 3 | 0010 | 12 |
| 4 | 0100 | 12 |
| 5 | 0000 | 4 |
| 6 | 0001 | 0 |
| 7 | 0011 | 0 |
| 8 | 0101 | 0 |
| 9 | 0111 | 0 |
| 10 | 1000 | 0 |
| ... | ... | ... |

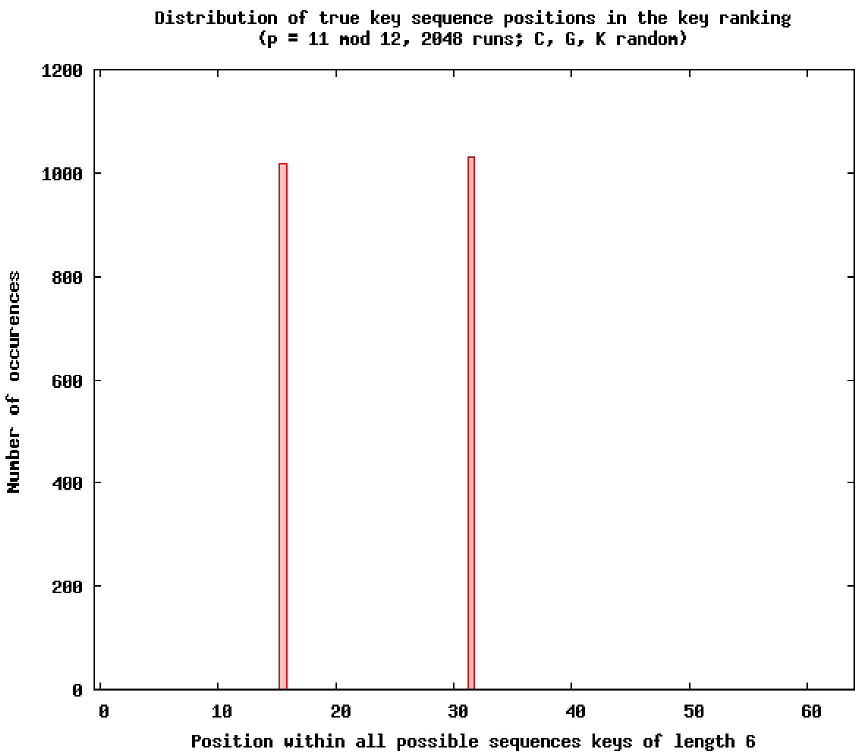
# TSPE on Random Curves

DEMO4




$p \equiv 7 \pmod{12}$, $\ell = 6$

Average position: $\mu_{pos} = 5.10$

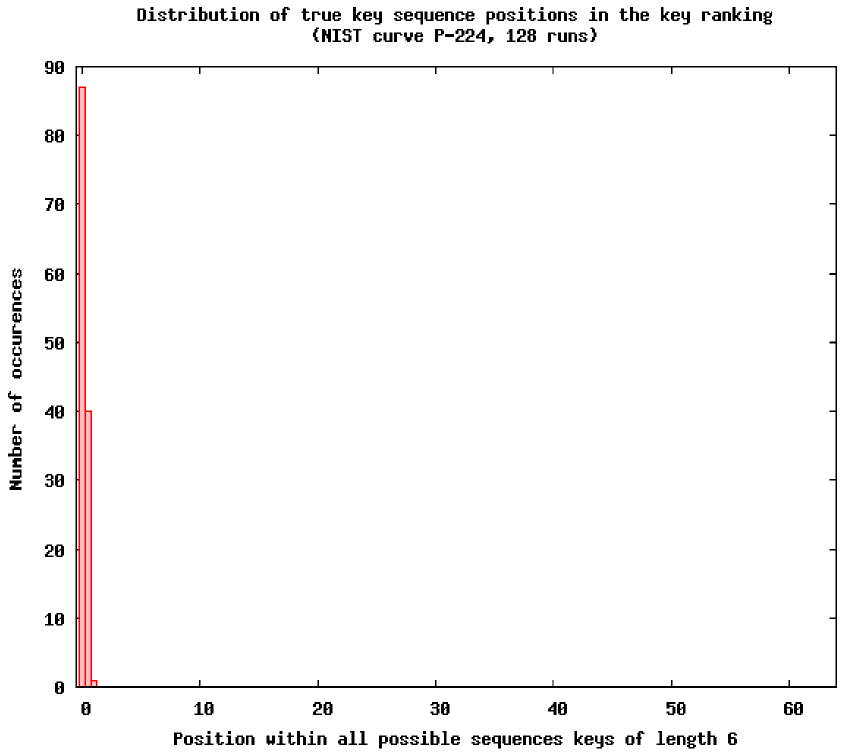


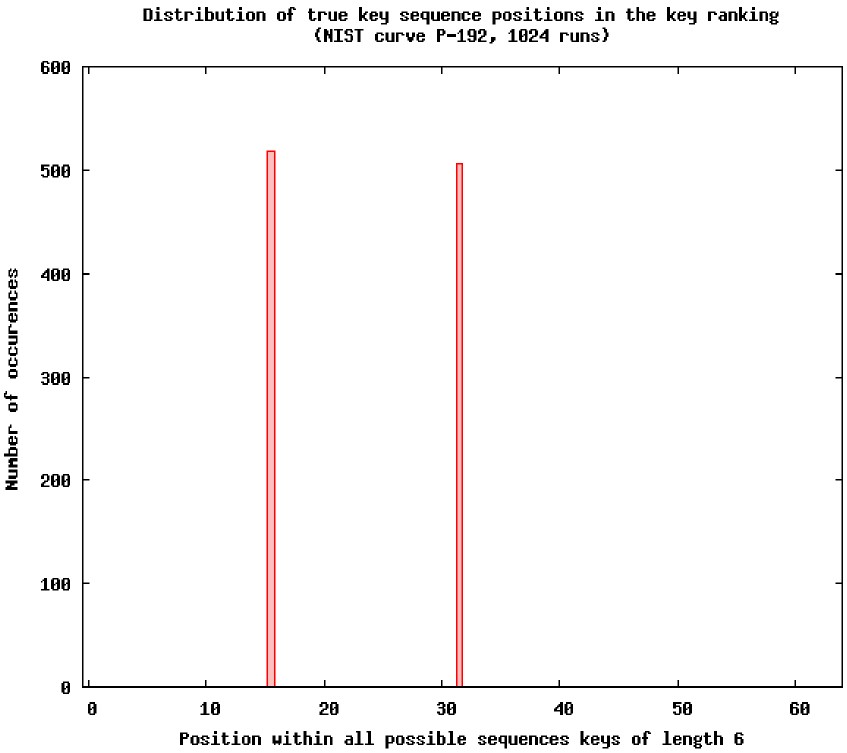

$p \equiv 11 \pmod{12}$, $\ell = 6$

Average position: $\mu_{pos} = 23.55$

# TSPE on NIST Curves

DEMO$_5$




**P-224**: $p \equiv 7 \pmod{12}$, $\ell = 6$

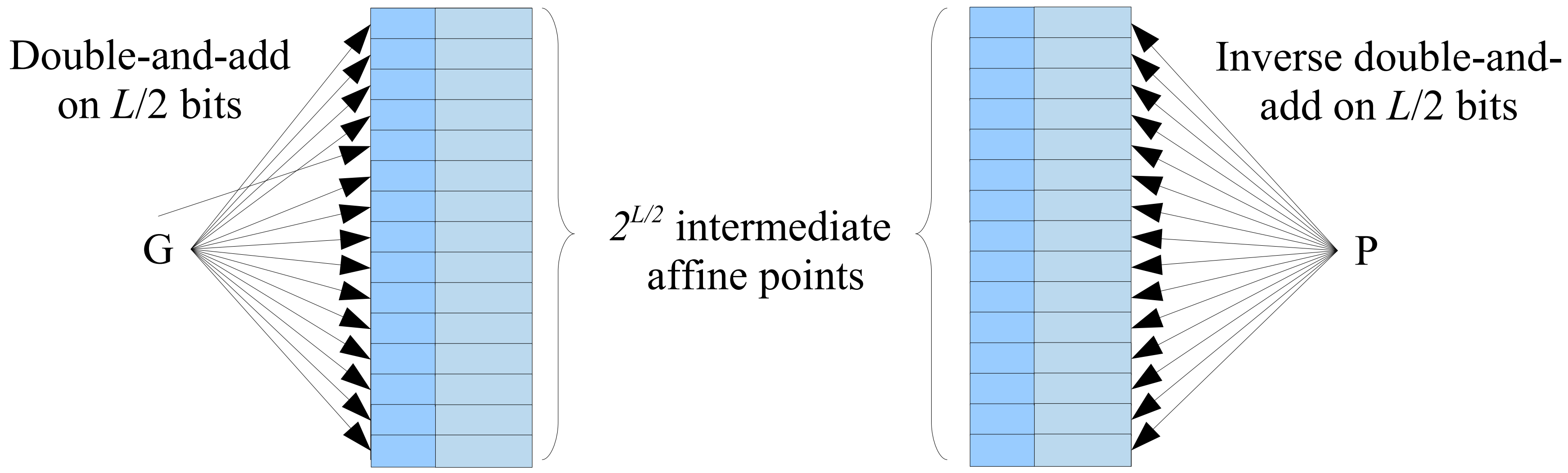
Average position: $\mu_{pos} = 0.16$

**P-192**: $p \equiv 11 \pmod{12}$, $\ell = 6$

Average position: $\mu_{pos} = 23.75$

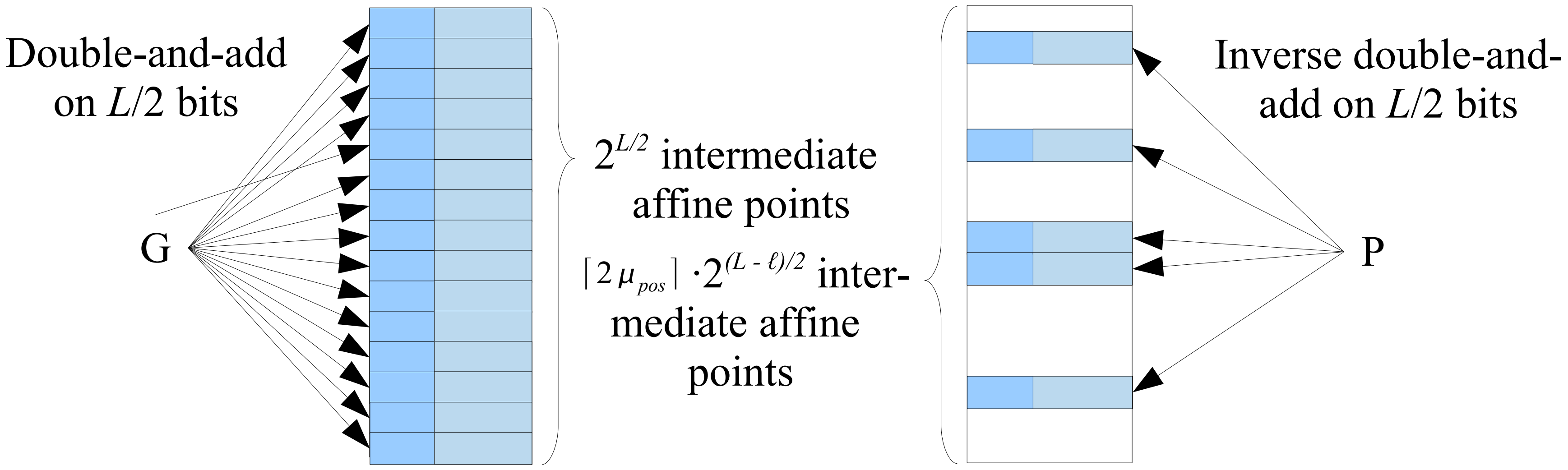
# Meet-in-the-Middle Attack

- **Time-Memory tradeoff:** Meet-in-the-middle Attack
  - Applicable to the double-and-add multiplication on affine points, e.g., with keys of size $L$
  - Complexity: $O(2^{\frac{L}{2}})$ in computation and $O(2^{\frac{L}{2}})$ in memory



# Meet-in-the-Middle Attack with TSPE

- **Probabilistic set of sequences of length $\ell$:** $\lceil 2\mu_{pos} \rceil \ll 2^{\frac{\ell}{2}}$
  - Keep keys with tailing sequences in the key ranking with positions $0 \le \text{index} \le \lceil 2\mu_{pos} \rceil$
  - Complexity: left side remains, right side: $O\left(\lceil 2\mu_{pos} \rceil 2^{\frac{L-\ell}{2}}\right)$



# Thwarting the Attack

- **Replace the double-and-add multiplication output**
  - randomize $(X : \varepsilon Y : \varepsilon Z)$ where $\varepsilon = \pm 1$
  - replace $(X : Y : Z)$ by $(\lambda^2 X : \lambda^3 Y : \lambda Z)$ with random $\lambda$ (more drastic)
- **Transforming the base point $G$**
  - randomly chosen projective representation
  - use a new representation for $G$ for every addition operation
- **Choice of the curve**
  - avoid curves such as P-224, use $p \equiv 11 \pmod{12}$ instead

# Conclusion

- **Projective representation of $P = [k]G$ leaks information**
  - depending on the characteristic of the underlying field
  - $p \equiv 1, 7 \pmod{12}$ allow to obtain more information (pre-images)
- **Key rankings and true sequence position expectations**
  - reduce the workload in brute-force attacks/time-memory tradeoffs
  - in particular the NIST curve P-224 is prone to the attacks
- **Thwarting the attacks**
  - through careful implementation and choice of the curve

# Questions

Thank you for your attention.

# Demonstrations: Full Scenario

DEMO

- Encrypt a key on P-256

  `./examples_encryption.sh` Scenario 3

- Find the TSPE set

  `./examples_find_TSPE_set.sh` Scenario 3

- Find the last bits

  `./examples_find_lastbits_AWK.sh` Scenario 3