#RSAC

RSAC 2025 Conference

Many Voices.
One Community.

SESSION ID: CRYP-W01

Hardware and Software Implementations: Post-Quantum Online/Offline Signatures

James Howe

Staff Research Scientist
SandboxAQ
https://www.linkedin.com/in/jameshowe1729/

Outline

- Introduction
  - A refresher on Post-Quantum Cryptography
  - What is the Online/Offline paradigm
  - Why could this intersection work?
- Our Contributions
- Results

RSAC 2025 Conference

Introduction

Why post-quantum cryptography matters

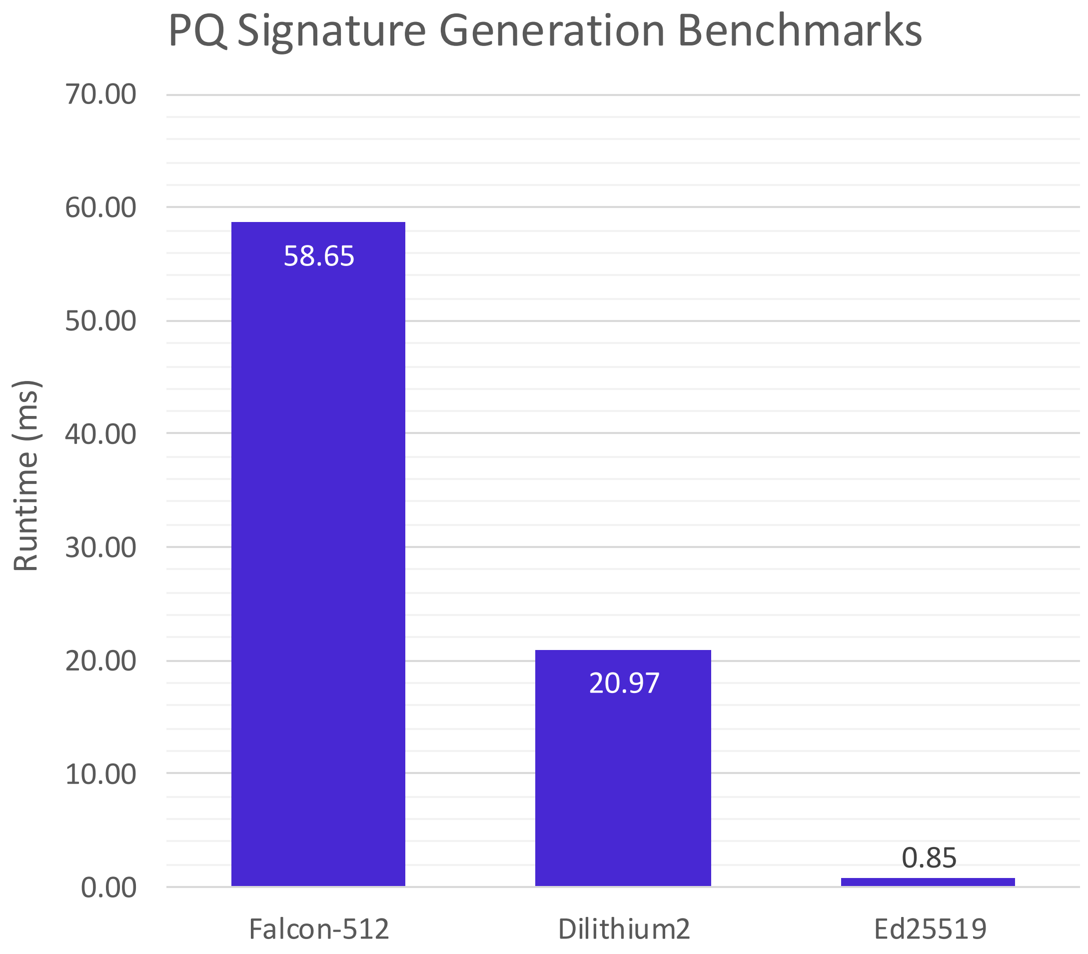
Many Voices.
One Community.

Why Post-Quantum Matters

The Quantum Threat	Practical quantum computers are coming RSA & ECC vulnerable via Shor's algorithm
NIST's PQC Efforts	Standardizing quantum-secure cryptosystems. FIPS standards: ML-KEM, ML-DSA (Dilithium), SLH-DSA, & Falcon.
PQC Signature Issues	High computational costs → Slower than classical signatures. Impacts smart cards, IoT, embedded devices.

PQ Signature Embedded Performance

- Why This Matters
  - Ed25519 is fast (used today in SSH, TLS, etc.).
  - Falcon & Dilithium are much slower due to heavy math operations.
- **Impact**: Real-time applications struggle with PQC signatures.
- **Solution**: Online/Offline signatures
  - Precompute heavy operations “offline” to make signing faster.



Introduction

What are online/offline signatures?

What are online/offline signatures?

- Introduced by Even, Goldreich, & Micali (1989, 1996)
- Allows efficient signing on devices with limited computational power.
  - 💳 Smart Cards – auth for ID cards, secure access, etc.
  - 🏪 POS (Point-of-Sale) Payment Systems – NFC & RFID-based contactless payments.
  - 🔐 Secure Transactions – Used in low-power devices where fast signing is required.

On-Line/Off-Line Digital Signatures*

Shimon Even
Computer Science Department, Technion—Israel Institute of Technology, Haifa 32000, Israel
even@cs.technion.ac.il

Oded Goldreich
Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel
oded@wisdom.weizmann.ac.il

Silvio Micali
Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, U.S.A.
silvio@theory.lcs.mit.edu

Communicated by Gilles Brassard

Received 19 August 1992 and revised 21 December 1994

Abstract. A new type of signature scheme is proposed. It consists of two phases. The first phase is performed off-line, before the message to be signed is even known. The second phase is performed on-line, once the message to be signed is known, and is supposed to be very fast. A method for constructing such on-line/off-line signature schemes is presented. The method uses one-time signature schemes, which are very fast, for the on-line signing. An ordinary signature scheme is used for the off-line stage.

In a practical implementation of our scheme, we use a variant of Rabin's signature scheme (based on factoring) and DES. In the on-line phase all we use is a moderate amount of DES computation and a single modular multiplication. We stress that the costly modular exponentiation operation is performed off-line. This implementation is ideally suited for electronic wallets or smart cards.

Key words. Digital signatures, Integer factorization, RSA, DES, One-time signature schemes, Error-correcting codes, Chosen message attack.

What are online/offline signatures?

- Splits signing into two phases:
  - Offline Phase 🛠️ → Precompute expensive operations before signing (e.g., modular exponentiation)
  - Online Phase ⚡ → Fast signing when needed using precomputed data

On-Line/Off-Line Digital Signatures*

Shimon Even
Computer Science Department, Technion—Israel Institute of Technology, Haifa 32000, Israel
even@cs.technion.ac.il

Oded Goldreich
Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel
oded@wisdom.weizmann.ac.il

Silvio Micali
Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, U.S.A.
silvio@theory.lcs.mit.edu

Communicated by Gilles Brassard

Received 19 August 1992 and revised 21 December 1994

Abstract. A new type of signature scheme is proposed. It consists of two phases. The first phase is performed off-line, before the message to be signed is even known. The second phase is performed on-line, once the message to be signed is known, and is supposed to be very fast. A method for constructing such on-line/off-line signature schemes is presented. The method uses one-time signature schemes, which are very fast, for the on-line signing. An ordinary signature scheme is used for the off-line stage.

In a practical implementation of our scheme, we use a variant of Rabin's signature scheme (based on factoring) and DES. In the on-line phase all we use is a moderate amount of DES computation and a single modular multiplication. We stress that the costly modular exponentiation operation is performed off-line. This implementation is ideally suited for electronic wallets or smart cards.

Key words. Digital signatures, Integer factorization, RSA, DES, One-time signature schemes, Error-correcting codes, Chosen message attack.

What are online/offline signatures?

- Offline precomputation phase 🛠️
  - Using a regular signature scheme, generates sign/verify long-term keys
  - Generates one-time sign/verify keys
  - Signs the one-time verify key with long term key
- Online Phase ⚡
  - Quickly verifies the one-time key
  - Signs the message/challenge with the one-time key
  - Quickly verifies the one-time signature created
- ⚠️ We are verifying twice, only works if 2*verifications are much faster than one regular signing

On-Line/Off-Line Digital Signatures*

Shimon Even
Computer Science Department, Technion—Israel Institute of Technology, Haifa 32000, Israel
even@cs.technion.ac.il

Oded Goldreich
Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel
oded@wisdom.weizmann.ac.il

Silvio Micali
Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, U.S.A.
silvio@theory.lcs.mit.edu

Communicated by Gilles Brassard

Received 19 August 1992 and revised 21 December 1994

Abstract. A new type of signature scheme is proposed. It consists of two phases. The first phase is performed off-line, before the message to be signed is even known. The second phase is performed on-line, once the message to be signed is known, and is supposed to be very fast. A method for constructing such on-line/off-line signature schemes is presented. The method uses one-time signature schemes, which are very fast, for the on-line signing. An ordinary signature scheme is used for the off-line stage.

In a practical implementation of our scheme, we use a variant of Rabin's signature scheme (based on factoring) and DES. In the on-line phase all we use is a moderate amount of DES computation and a single modular multiplication. We stress that the costly modular exponentiation operation is performed off-line. This implementation is ideally suited for electronic wallets or smart cards.

Key words. Digital signatures, Integer factorization, RSA, DES, One-time signature schemes, Error-correcting codes, Chosen message attack.

RSAC 2025 Conference

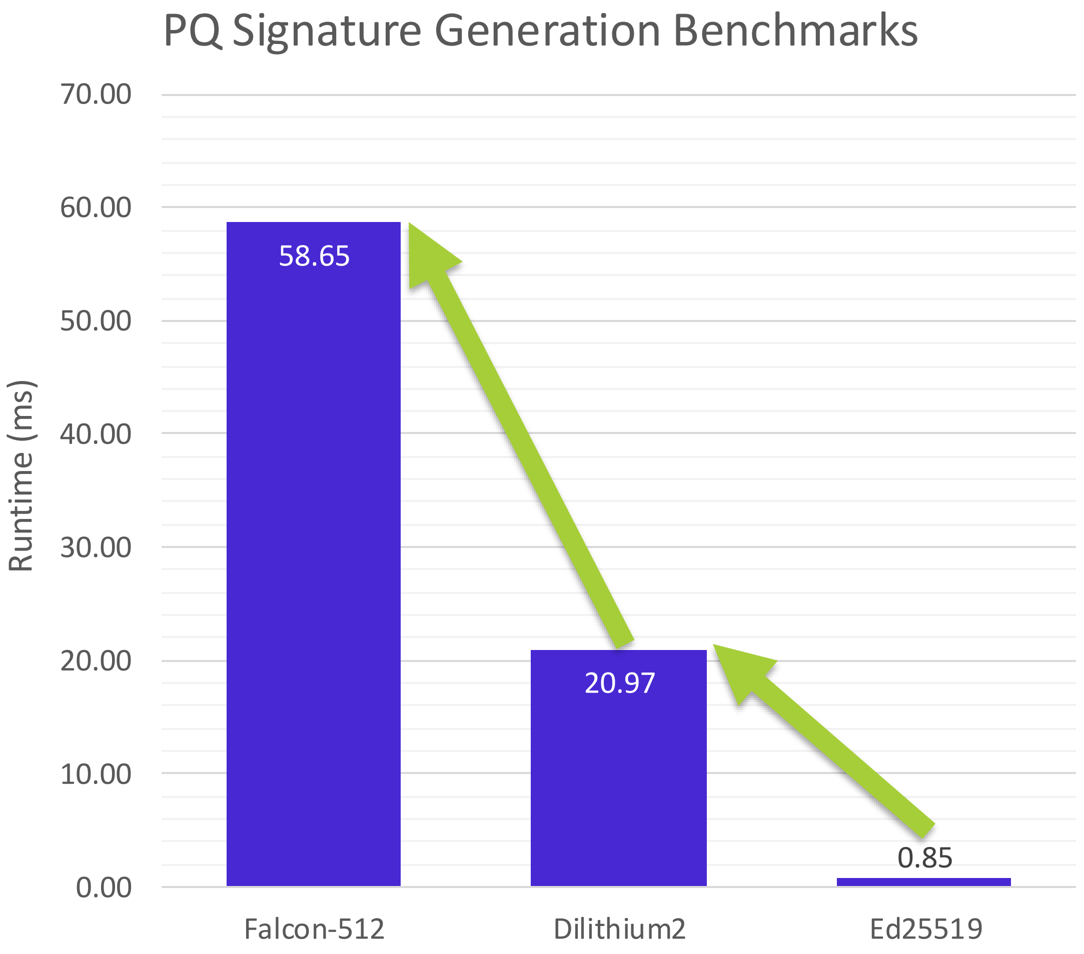
Post-Quantum Online/Offline Signatures

Why Falcon?

Many Voices.
One Community.

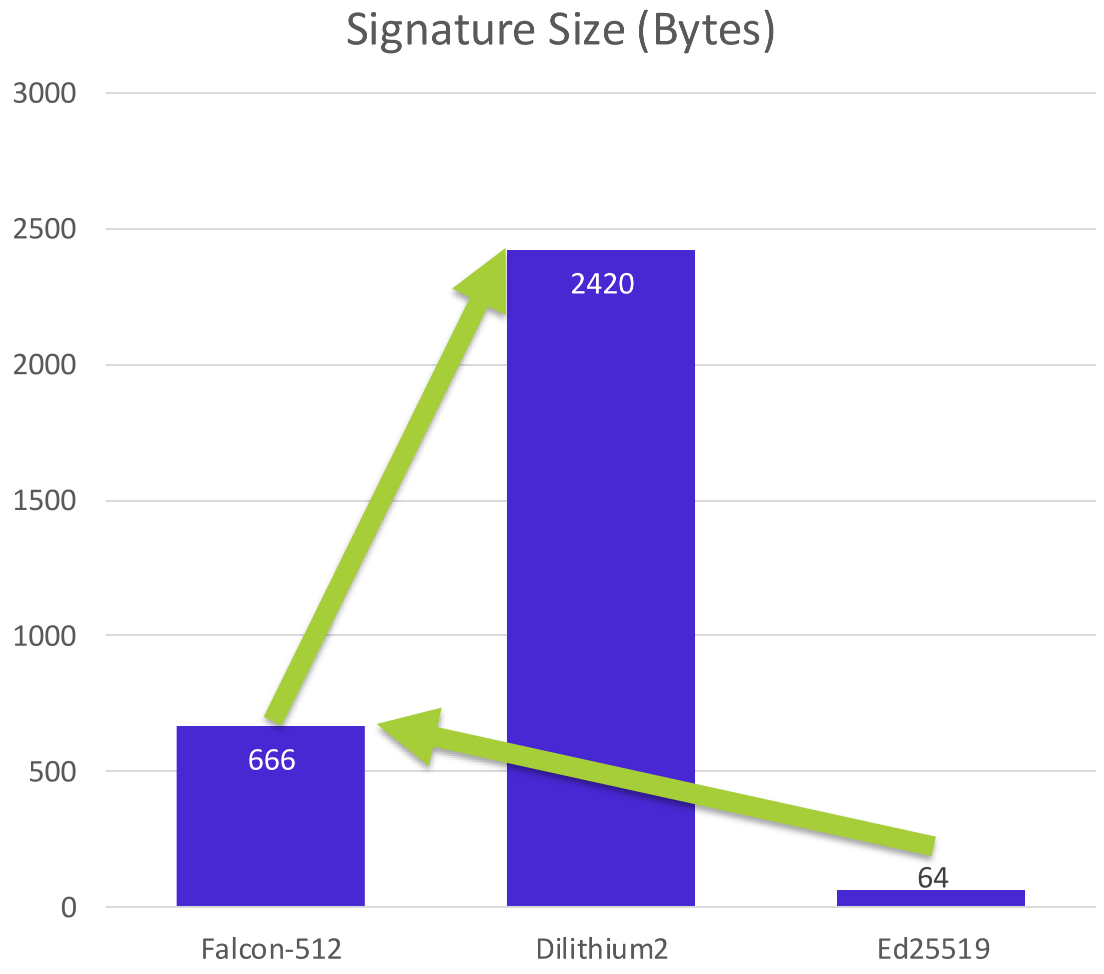
Why Falcon?

- At this point you may be wondering why not Dilithium?
  - Dilithium is much faster than Falcon!
  - There’s an order of magnitude between them!
  - Surely, it's a better choice for online/offline?



Why Falcon?

- Falcon more suited to small devices:
  - Compact Signatures 📏 – Falcon is 3.5x smaller
  - Fast Verification ⚡– Falcon verifies 3x faster
  - Falcon is also a NIST PQC standard
- Our goal:
  - Apply Online/Offline to Falcon's slower signing times!
  - Retain Falcon's advantages
  - Make POS etc. PQ and practical!



Falcon: Under the Hood 🚘

Why is Falcon so expensive to sign?

- Many *FFT* conversions of the secret key
  - Costs more than total Dilithium2 signing
  - And this happens twice!
- *ffSampling* – trapdoor Gaussian sampler
  - It's recursive, uses a lot of randomness, and is computationally expensive
- Falcon requires *floating-point* operations
  - And on small embedded devices with no FPU means costly emulation

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\$\ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

  do

    $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T});$

    $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}};$

  while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

  $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

  $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\!\$\ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

    do

        $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$;

        $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}$;

    while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

    $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

    $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs more than total Dilithium2 signing
  - And this happens twice!

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\!\$ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

  do

    $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$;

    $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}$;

  while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

  $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

  $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs igning
  - And t

Offline

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$$
\begin{array}{l}
\mathbf{r} \leftarrow\!\!\$\ \{0,1\}^{320} \\
c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n) \\
\mathbf{t} \leftarrow \left(-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f)\right) \\
\quad \textbf{do} \\
\qquad \textbf{do} \\
\qquad\quad \mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T}); \\
\qquad\quad \mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}; \\
\qquad \textbf{while } \|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor \\
\qquad (s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s}) \\
\qquad s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328) \\
\textbf{while } s = \perp \\
\textbf{return } \sigma := (\mathbf{r}, s)
\end{array}
$$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs … igning
  - And t…
- *ffSampling* – trapdoor Gaussian sampler
  - It's recursive, uses a lot of randomness, and is computationally expensive

Offline

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\$\; \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

  do

    $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$;

    $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}$;

  while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

  $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

  $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs … igning
  - And t…
- *ffSampling* – trapdoor Gaussian sampler
  - It's recu… ess, and is c…

Offline

Lazify

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\!\$ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

  do

    $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$;

    $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}$;

  while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

  $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

  $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs ... igning
  - And t
- *ffSampling* – trapdoor Gaussian sampler
  - It's recu ... ess, and is c

Offline

Lazify

Falcon requires *floating-point* operations

And on small embedded devices with no FPU means costly emulation

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\!\$ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

  do

    $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$;

    $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}$;

  while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

  $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

  $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs … igning
  - And t…
- *ffSampling* – trapdoor Gaussian sampler
  - It's recu … ess, and is c…

Offline

Lazify

Falcon requires *floating-point* operations

And on smal… FPU means …

Sign(sk, μ, $\lfloor \beta^2 \rfloor$)

$r \leftarrow\!\!\$ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(r \| \mu, q, n)$

$t \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

  do

    $z \leftarrow \mathsf{ffSampling}_n(t, \mathsf{T})$;

    $s = (t - z) \odot \hat{\mathbf{B}}$;

  while $\|s\|^2 > \lfloor \beta^2 \rfloor$

  $(s_0, s_1) \leftarrow \mathsf{invFFT}(s)$

  $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (r, s)$

RSAC 2025 Conference

Post-Quantum Online/Offline Signatures

Lazy Falcon “Few Times” Signatures

Many Voices.
One Community.

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs more than total Dilithium2 signing
  - And this happens twice!
- *ffSampling* – trapdoor Gaussian sampler
  - It's recursive, uses a lot of randomness, expensive

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\!\$ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

    do

        $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$;

        $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}$;

    while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

    $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

    $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \perp$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs more than total Dilithium2 signing
  - And this happens twice!
- *ffSampling* – trapdoor Gaussian sampler
  - It's recursive, uses a lot of randomness, expensive

$\mathsf{PreSign}(\mathsf{sk})$

$\mathbf{B} \leftarrow [g, -f; G, -F]$

$\hat{\mathbf{B}} \leftarrow \mathsf{FFT}(\mathbf{B})$

$u_0, u_1 \leftarrow \mathcal{D}_z^2$

$c_{pre} \leftarrow \mathsf{ComputeTarget}(h, u_0, u_1)$

$\mathbf{return}\ (\rho := u_0, u_1, c_{pre})$

$\mathsf{ComputeTarget}(h, u_0, u_1)$

$\hat{u_1} \leftarrow \mathsf{NTT}(u_1)$

$\hat{t} \leftarrow h \odot \hat{u_1}$

$t \leftarrow \mathsf{invNTT}(\hat{t}) + u_0$

$\mathbf{return}\ t$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs more than total Dilithium2 signing
  - And this happens twice!

- *ffSampling* – trapdoor Gaussian sampler
  - It's recursive, uses a lot of randomness, expensive

$\mathsf{Sign}(\mathsf{sk}, \mu, \lfloor \beta^2 \rfloor)$

$\mathbf{r} \leftarrow\!\!\$\ \{0,1\}^{320}$

$c \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$\mathbf{t} \leftarrow (-\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(F), -\frac{1}{q}\mathsf{FFT}(c) \odot \mathsf{FFT}(f))$

do

  do

    $\mathbf{z} \leftarrow \mathsf{ffSampling}_n(\mathbf{t}, \mathsf{T})$;

    $\mathbf{s} = (\mathbf{t} - \mathbf{z}) \odot \hat{\mathbf{B}}$;

  while $\|\mathbf{s}\|^2 > \lfloor \beta^2 \rfloor$

  $(s_0, s_1) \leftarrow \mathsf{invFFT}(\mathbf{s})$

  $s \leftarrow \mathsf{Compress}(s_1, 8 \cdot \mathsf{sbytelen} - 328)$

while $s = \bot$

return $\sigma := (\mathbf{r}, s)$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs more than total Dilithium2 signing
  - And this happens twice!

- *ffSampling* – trapdoor Gaussian sampler
  - It's recursive, uses a lot of randomness, expensive

$\mathsf{Sign}(\mathsf{sk}, \rho, \mu)$

$\mathbf{r} \leftarrow\!\!\$ \{0,1\}^{320}$

$c' \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$c \leftarrow c' + c_{\mathrm{pre}}$

$s_0', s_1' \leftarrow \mathsf{SampPre}_{\gg}(c, \hat{\mathbf{B}})$

$s_0 \leftarrow s_0' - u_0,\ s_1 \leftarrow s_1' - u_1$

$s \leftarrow \mathsf{Compress}(s_1)$

$\mathbf{return}\ \sigma := (\mathbf{r}, s)$

$\mathsf{SampPre}_{\gg}(c, \hat{\mathbf{B}})$

$\hat{c} \leftarrow \mathsf{FFT}(c)$

$/\!/ \ \hat{\mathbf{t}} = (\mathsf{FFT}(c), \mathsf{FFT}(0)) \cdot \hat{\mathbf{B}}^{-1}$

$\hat{\mathbf{t}} \leftarrow \left(-\frac{1}{q} \cdot \hat{c} \odot \mathsf{FFT}(F), -\frac{1}{q} \cdot \hat{c} \odot \mathsf{FFT}(f)\right)$

$\mathbf{t} \leftarrow \mathsf{invFFT}(\hat{\mathbf{t}});\quad \mathbf{t} \leftarrow \mathsf{round}(\mathbf{t});\quad \hat{\mathbf{t}} \leftarrow \mathsf{FFT}(\mathbf{t})$

$\hat{\mathbf{s}} \leftarrow (\mathsf{FFT}(c), \mathsf{FFT}(0)) - \hat{\mathbf{t}} \odot \hat{\mathbf{B}}$

$\mathbf{s} \leftarrow \mathsf{invFFT}(\hat{\mathbf{s}})$

$\mathbf{return}\ \mathbf{s}$

Lazy Falcon

Our Contributions

- Many *FFT* conversions of the secret key
  - Costs more than total Dilithium2 signing
  - And this happens twice!

- *ffSampling* – trapdoor Gaussian sampler
  - It's recursive, uses a lot of randomness, expensive

$\underline{\mathsf{Sign}(\mathsf{sk}, \rho, \mu)}$

$\mathbf{r} \leftarrow\!\!\$\ \{0,1\}^{320}$

$c' \leftarrow \mathsf{HashToPoint}(\mathbf{r} \| \mu, q, n)$

$c \leftarrow c' + c_{\mathrm{pre}}$

$s_0', s_1' \leftarrow \mathsf{SampPre}_{\gg}(c, \hat{\mathbf{B}})$

$s_0 \leftarrow s_0' - u_0,\ s_1 \leftarrow s_1' - u_1$

$s \leftarrow \mathsf{Compress}(s_1)$

$\textbf{return}\ \sigma := (\mathbf{r}, s)$

$\underline{\mathsf{SampPre}_{\gg}(c, \hat{\mathbf{B}})}$

$\hat{c} \leftarrow \mathsf{FFT}(c)$

$/\!/\ \hat{\mathbf{t}} = (\mathsf{FFT}(c), \mathsf{FFT}(0)) \cdot \hat{\mathbf{B}}^{-1}$

$\hat{\mathbf{t}} \leftarrow \left(-\frac{1}{q} \cdot \hat{c} \odot \mathsf{FFT}(F), -\frac{1}{q} \cdot \hat{c} \odot \mathsf{FFT}(f)\right)$

$\mathbf{t} \leftarrow \mathsf{invFFT}(\hat{\mathbf{t}});\ \boxed{\mathbf{t} \leftarrow \mathsf{round}(\mathbf{t});}\ \hat{\mathbf{t}} \leftarrow \mathsf{FFT}(\mathbf{t})$

$\hat{\mathbf{s}} \leftarrow (\mathsf{FFT}(c), \mathsf{FFT}(0)) - \hat{\mathbf{t}} \odot \hat{\mathbf{B}}$

$\mathbf{s} \leftarrow \mathsf{invFFT}(\hat{\mathbf{s}})$

$\textbf{return}\ \mathbf{s}$

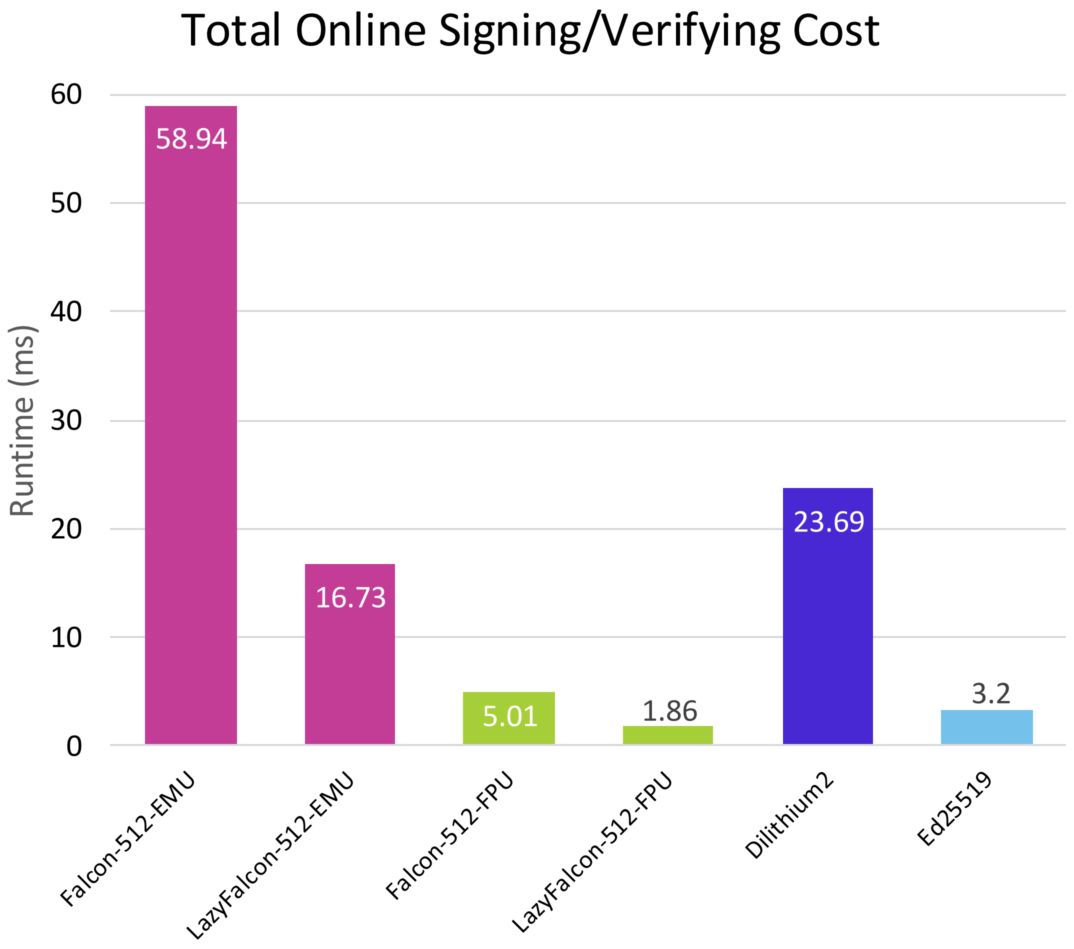
RSAC 2025 Conference

Benchmarking Post-Quantum Signatures

Many Voices.
One Community.

Benchmarking PQ signatures

- Pink is Falcon emulated
  – Most realistic for small/embedded
  – Shows a ~4x saving
- Green is Falcon using native FPU
  – Roughly the same savings vs Falcon
  – Nearly 2x faster than Ed25519
- Blue is Dilithium2
  – Much slower verify ~3 ms
  – Lazy Falcon at least 1.5x faster and nearly 13x faster with FPU



Benchmarked on Raspberry Pi 3B (ARM Cortex-A53), latest GCC

RSAC 2025 Conference

Post-Quantum Online/Offline Signatures

Lazy Falcon "Few Times" Signatures: Pros and Cons

Many Voices.
One Community.

Pros and Cons of Lazy Falcon

- Pros
  - We can sign/verify faster than Ed25519
  - Lazy Falcon is compatible with Falcon
    - Same key generation, verification has one difference
  - You *kinda* get *some* side-channel protection “for free”
- Cons
  - Sig. cost is slightly bigger, but smaller than ML-DSA/Dilithium
  - “Few-times” signature is realistically at most 4 or 5 times

RSAC 2025 Conference

Many Voices. **One Community.**

Thank you

More information:

- Martin R. Albrecht, Nicolas Gama, James Howe, and Anand K. Narayanan, 2025. *Post-Quantum Online/Offline Signatures*, eprint.iacr.org/2025/117
- Reference code: https://github.com/jameshoweee/online-offline-sigs