Escuela de Postgrado y Educación Continua
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

IV Simposio POSTGRADO 26
Ingeniería - Ciencias - Innovación

dcc CIENCIAS DE LA COMPUTACIÓN UNIVERSIDAD DE CHILE

GSkit: Un DSL/framework para Pruebas de Groth-Sahai

Benjamín Hurtado, Matías Toro, Federico Olmedo.
Ciencias de la Computación, Ciencia de Datos e Inteligencia Artificial

INTRODUCCIÓN Y PROBLEMA

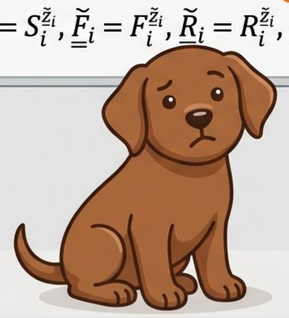
Las pruebas de Groth-Sahai son pruebas de conocimiento cero (*Zero Knowledge Proof*) no interactivas (NIZK/NIWI) eficientes y ampliamente usadas en firmas de grupo, credenciales anónimas, votación electrónica y firmas basadas en atributos.
Su construcción manual es laboriosa, de bajo nivel, propensa a errores y está fuertemente acoplada a primitivas de emparejamientos bilineales (*Bilinear Pairings*).
No existe una herramienta de alto nivel que abstraiga esta complejidad: cada ecuación se construye una vez para probar y, casi idénticamente, otra vez para verificar.

$$sk'_{id,\,a_i} = (R'_i, S'_i, \tilde{R}'_i)$$

$$sk_{id,\,a_i} := (R_i, \tilde{R}_i, S_i) = \left(R'_i \cdot G^{r'},\ (\tilde{R}'_i \cdot \tilde{H}^{r'}), S'_i \cdot R'^{2r'}_i \cdot G^{r'2}\right)$$

$$e(g^a, h^b) = e(g,h)^{ab} \quad ?$$

$$\underline{\tilde{S}}_i = S_i^{\tilde{z}_i}, \underline{\tilde{F}}_i = F_i^{\tilde{z}_i}, \underline{\tilde{R}}_i = R_i^{\tilde{z}_i}, \underline{\tilde{G}}_i = G_i^{\tilde{z}_i}$$



OBJETIVOS

Presentar GSkit, un DSL en Python para especificar pruebas de Groth-Sahai de forma declarativa.
Abstraer la complejidad matemática subyacente con una sintaxis cercana a la notación del esquema.
Detectar errores en tiempo de especificación, no en tiempo de ejecución, mediante una gramática estructurada y verificación de tipos.
Generar automáticamente, a partir de una única especificación, el código de generación y de verificación de la prueba.
Evaluar el enfoque en dos casos de estudio de complejidad creciente (BLS y ABS-UCL), comparándolo contra una implementación manual previa.

¿Qué es ZKP?

Un protocolo de Cero Conocimiento (*Zero-Knowledge Proof*) permite a un probador (Alicia) convencer a un verificador (Beto) de que conoce un secreto, o de que una afirmación es verdadera, sin revelar ninguna información adicional.
Tras un intercambio de desafío y respuesta, Beto queda convencido… pero sigue sin tener ninguna pista sobre cuál es el secreto.
Groth-Sahai da un paso más: es **no interactivo**. En vez de una conversación de desafíos y respuestas, Alicia y Beto ya comparten de antemano una misma "regla de juego" pública (*Common Reference String*).
Debe cumplir completitud, solidez (*soundness*) y conocimiento cero/indistinguibilidad de testigos (NIZK/NIWI).

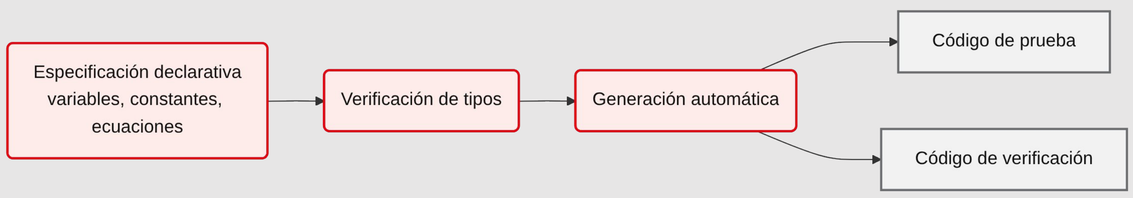
METODOLOGÍA: GSkit

El desarrollador anota su código de verificación con el decorador `@gsfy` y declara **variables**, **constantes** y **ecuaciones** del esquema.

Un sistema de **tipado fuerte y estático** valida que la especificación esté bien formada antes de construir la prueba.

Traslada una clase de errores desde el tiempo de ejecución al tiempo de especificación.

```
@gsfy(...)
def absucl_spec(..., a_i, ...):
  R_i = constant(sk_id_a_i[0])
  Ftilda = variable(Ftilda)
  ...
GS_STRING
```



RESULTADOS

BLS: se extrae una única ecuación de producto de emparejamientos desde la rutina de verificación de firma, con mínimo esfuerzo.

```
@gsfy
def verify(self, vk, m, signature):
    g2 = self.g2
    lhs = signature.pair(g2)
    rhs = G1Element.hash_from_string(m).pair(vk)
    GS_STRING = """
        variables: signature : G1
        constants: g2 : G2 , rhs : GT
        equations: signature * g2 = rhs
    """
return lhs == rhs
```

ABS-UCL: esquema compuesto por varios subprotocolos que comparten ecuaciones (política MSP, token de enlace, firmas de atributos aleatorizadas, pseudo-atributo).

- Manualmente: 85 líneas en `sign()` + 84 casi idénticas en `verify()` = 169 líneas duplicadas, con 60 objetos construidos a mano. Con GSkit: 8 ecuaciones declarativas escritas **una sola vez**.
- El total de líneas no es la métrica relevante (de hecho sube de 420 a 540, pues cada subprotocolo suma su propia especificación); lo que se elimina es el código de ecuaciones duplicado, que es donde ocurren los errores.

CONCLUSIONES

GSkit simplifica la construcción de pruebas de Groth-Sahai para desarrolladores.
Reduce el esfuerzo y la probabilidad de error sin sacrificar la seguridad.
Facilita la adopción de ZKP complejas en aplicaciones criptográficas.
Traslada una clase de errores del tiempo de ejecución al tiempo de especificación, eliminando el código duplicado sin sacrificar la seguridad criptográfica.
Trabajo futuro: verificación por lotes, más ejemplos, e integración en una aplicación completa (el foro anónimo que motivó este trabajo).

REFERENCIAS

[1] J. Groth y A. Sahai, "Efficient non-interactive proof systems for bilinear groups," en Advances in Cryptology – EUROCRYPT 2008, pp. 415–432, 2008.
[2] D. Boneh, B. Lynn y H. Shacham, "Short signatures from the Weil pairing," en Advances in Cryptology – ASIACRYPT 2001, pp. 514–532, 2001.
[3] A. El Kaafarani y E. Ghadafi, "Attribute-based signatures with user-controlled linkability without random oracles," en IMACC 2017, pp. 161–184, 2017.