Gruppi di Ricerca ”De Cifris” a Bari

- **Università di Bari**. Algoritmi per la crittografia e la crittoanalisi.
  - Roberto LA SCALA - MAT/02
  - Vincenzo NARDOZZA - MAT/02
  - Pierluigi AMODIO - MAT/08
- **Politecnico di Bari**. Codici per la trasmissione su reti dati.
  - Angela AGUGLIA - MAT/03
  - Francesco PAVESE - MAT/03
- **Associazione ”Alumni Mathematica”**. Tecnologie Blockchain.

Gruppi di Ricerca ”De Cifris” a Bari

- **Università di Bari**. Algoritmi per la crittografia e la crittoanalisi.
  - Roberto LA SCALA - MAT/02
  - Vincenzo NARDOZZA - MAT/02
  - Pierluigi AMODIO - MAT/08
- **Politecnico di Bari**. Codici per la trasmissione su reti dati.
  - Angela AGUGLIA - MAT/03
  - Francesco PAVESE - MAT/03
- **Associazione ”Alumni Mathematica”**. Tecnologie Blockchain.

Gruppi di Ricerca ”De Cifris” a Bari

- **Università di Bari**. Algoritmi per la crittografia e la crittoanalisi.
  - Roberto LA SCALA - MAT/02
  - Vincenzo NARDOZZA - MAT/02
  - Pierluigi AMODIO - MAT/08
- **Politecnico di Bari**. Codici per la trasmissione su reti dati.
  - Angela AGUGLIA - MAT/03
  - Francesco PAVESE - MAT/03
- **Associazione ”Alumni Mathematica”**. Tecnologie Blockchain.

Gruppi di Ricerca ”De Cifris” a Bari

- **Università di Bari**. Algoritmi per la crittografia e la crittoanalisi.
  - Roberto LA SCALA - MAT/02
  - Vincenzo NARDOZZA - MAT/02
  - Pierluigi AMODIO - MAT/08
- **Politecnico di Bari**. Codici per la trasmissione su reti dati.
  - Angela AGUGLIA - MAT/03
  - Francesco PAVESE - MAT/03
- **Associazione ”Alumni Mathematica”**. Tecnologie Blockchain.

- **Corsi universitari**
  - Crittografia (Laurea Magistrale in Matematica)
  - Cryptography (Laurea Magistrale in Ingegneria delle Telecomunicazioni)
- **Tesi, stages e tirocini formativi**
- **Editoria scientifica**
  - Mediterranean Journal of Mathematics (1.181 Impact Factor 2018), Massimiliano Sala editor per la Crittografia.
- **Iniziative di divulgazione**
  - Notte europea della Ricerca
  - Museo della Matematica
  - Seminari criptovalute (Alumni)

- **Corsi universitari**
  - Crittografia (Laurea Magistrale in Matematica)
  - Cryptography (Laurea Magistrale in Ingegneria delle Telecomunicazioni)
- **Tesi, stages e tirocini formativi**
- **Editoria scientifica**
  - Mediterranean Journal of Mathematics (1.181 Impact Factor 2018), Massimiliano Sala editor per la Crittografia.
- **Iniziative di divulgazione**
  - Notte europea della Ricerca
  - Museo della Matematica
  - Seminari criptovalute (Alumni)

- **Corsi universitari**
  - Crittografia (Laurea Magistrale in Matematica)
  - Cryptography (Laurea Magistrale in Ingegneria delle Telecomunicazioni)
- **Tesi, stages e tirocini formativi**
- **Editoria scientifica**
  - Mediterranean Journal of Mathematics (1.181 Impact Factor 2018), Massimiliano Sala editor per la Crittografia.
- **Iniziative di divulgazione**
  - Notte europea della Ricerca
  - Museo della Matematica
  - Seminari criptovalute (Alumni)

- **Corsi universitari**
  - Crittografia (Laurea Magistrale in Matematica)
  - Cryptography (Laurea Magistrale in Ingegneria delle Telecomunicazioni)
- **Tesi, stages e tirocini formativi**
- **Editoria scientifica**
  - Mediterranean Journal of Mathematics (1.181 Impact Factor 2018), Massimiliano Sala editor per la Crittografia.
- **Iniziative di divulgazione**
  - Notte europea della Ricerca
  - Museo della Matematica
  - Seminari criptovalute (Alumni)

- Il tema principale dell'unità UNIBA sono i **cifrari a flusso**.
- Ampio utilizzo:
  - Keystream per reti wireless, internet, mobile;
  - Numeri pseudorandom, OTP;
  - Cifrari simmetrici, etc.
- Facilità di implementazione hardware/software, efficienza cifratura.
- Sicurezza analizzabile mediante "Statistical Test Suite" (NIST), crittoanalisi, etc.

- Il tema principale dell'unità UNIBA sono i **cifrari a flusso**.
- Ampio utilizzo:
  - Keystream per reti wireless, internet, mobile;
  - Numeri pseudorandom, OTP;
  - Cifrari simmetrici, etc.
- Facilità di implementazione hardware/software, efficienza cifratura.
- Sicurezza analizzabile mediante "Statistical Test Suite" (NIST), crittoanalisi, etc.

- Il tema principale dell'unità UNIBA sono i **cifrari a flusso**.
- Ampio utilizzo:
  - Keystream per reti wireless, internet, mobile;
  - Numeri pseudorandom, OTP;
  - Cifrari simmetrici, etc.
- Facilità di implementazione hardware/software, efficienza cifratura.
- Sicurezza analizzabile mediante "Statistical Test Suite" (NIST), crittoanalisi, etc.

- Il tema principale dell'unità UNIBA sono i **cifrari a flusso**.
- Ampio utilizzo:
  - Keystream per reti wireless, internet, mobile;
  - Numeri pseudorandom, OTP;
  - Cifrari simmetrici, etc.
- Facilità di implementazione hardware/software, efficienza cifratura.
- Sicurezza analizzabile mediante "Statistical Test Suite" (NIST), crittoanalisi, etc.

- I cifrari a flusso sono essenzialmente funzioni del tempo (clock) a valori in uno spazio di vettori di bit (stati), diciamo $f : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$.
- Il keystream è costituito in genere da una singola coordinata di questi vettori.
- L'azione del keystream sul plaintext-stream è semplicemente la somma bit a bit (XOR).
- Queste funzioni sono computatibili ovvero definite ricorsivamente a partire da uno stato iniziale.
- La chiave del cifrario è in genere il suo stato iniziale.
- Per proteggere la chiave, si usa fornire il keystream solo da un certo clock in poi.

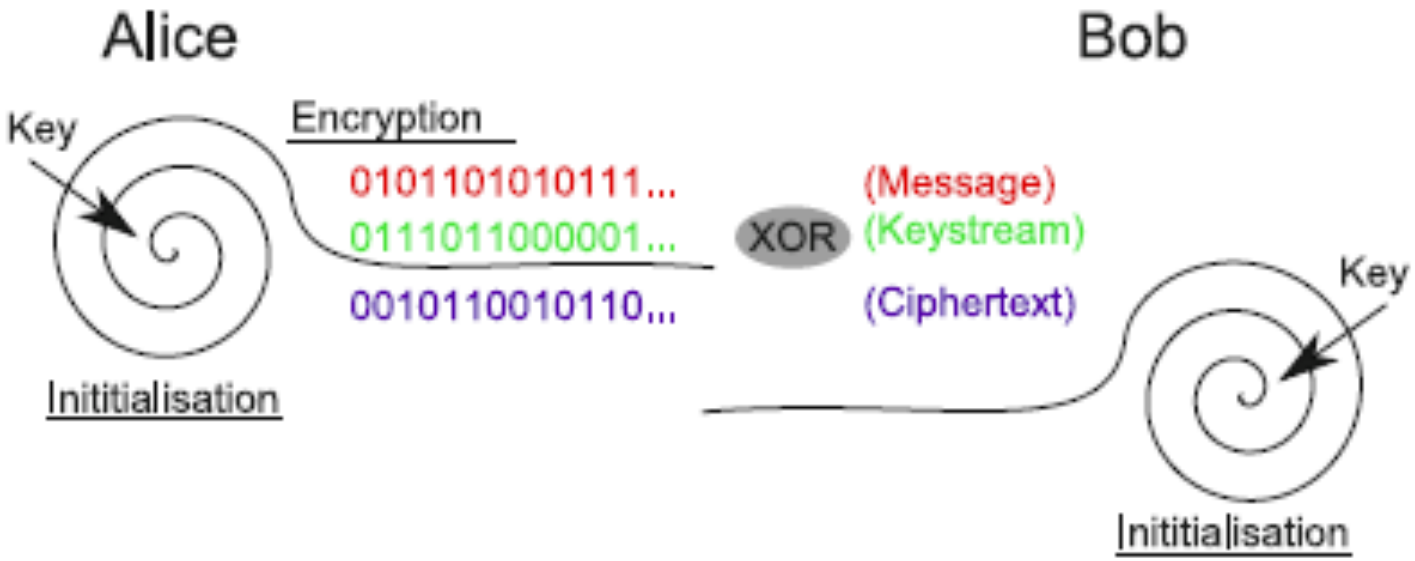
- I cifrari a flusso sono essenzialmente funzioni del tempo (clock) a valori in uno spazio di vettori di bit (stati), diciamo $f : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$.
- Il keystream è costituito in genere da una singola coordinata di questi vettori.
- L'azione del keystream sul plaintext-stream è semplicemente la somma bit a bit (XOR).
- Queste funzioni sono computatibili ovvero definite ricorsivamente a partire da uno stato iniziale.
- La chiave del cifrario è in genere il suo stato iniziale.
- Per proteggere la chiave, si usa fornire il keystream solo da un certo clock in poi.

- I cifrari a flusso sono essenzialmente funzioni del tempo (clock) a valori in uno spazio di vettori di bit (stati), diciamo $f : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$.
- Il keystream è costituito in genere da una singola coordinata di questi vettori.
- L'azione del keystream sul plaintext-stream è semplicemente la somma bit a bit (XOR).
- Queste funzioni sono computatibili ovvero definite ricorsivamente a partire da uno stato iniziale.
- La chiave del cifrario è in genere il suo stato iniziale.
- Per proteggere la chiave, si usa fornire il keystream solo da un certo clock in poi.

- I cifrari a flusso sono essenzialmente funzioni del tempo (clock) a valori in uno spazio di vettori di bit (stati), diciamo $f : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$.
- Il keystream è costituito in genere da una singola coordinata di questi vettori.
- L'azione del keystream sul plaintext-stream è semplicemente la somma bit a bit (XOR).
- Queste funzioni sono computatibili ovvero definite ricorsivamente a partire da uno stato iniziale.
- La chiave del cifrario è in genere il suo stato iniziale.
- Per proteggere la chiave, si usa fornire il keystream solo da un certo clock in poi.

- I cifrari a flusso sono essenzialmente funzioni del tempo (clock) a valori in uno spazio di vettori di bit (stati), diciamo $f : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$.
- Il keystream è costituito in genere da una singola coordinata di questi vettori.
- L'azione del keystream sul plaintext-stream è semplicemente la somma bit a bit (XOR).
- Queste funzioni sono computatibili ovvero definite ricorsivamente a partire da uno stato iniziale.
- La chiave del cifrario è in genere il suo stato iniziale.
- Per proteggere la chiave, si usa fornire il keystream solo da un certo clock in poi.

- I cifrari a flusso sono essenzialmente funzioni del tempo (clock) a valori in uno spazio di vettori di bit (stati), diciamo $f : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$.
- Il keystream è costituito in genere da una singola coordinata di questi vettori.
- L'azione del keystream sul plaintext-stream è semplicemente la somma bit a bit (XOR).
- Queste funzioni sono computatibili ovvero definite ricorsivamente a partire da uno stato iniziale.
- La chiave del cifrario è in genere il suo stato iniziale.
- Per proteggere la chiave, si usa fornire il keystream solo da un certo clock in poi.


Alice
Bob
Key
Encryption
01011010101111...
(Message)
01110110000001...
XOR
(Keystream)
00101100101100...
(Ciphertext)
Key
Initialisation
Initialisation

- La ricorsione che definisce molti cifrari a flusso largamente utilizzati è di fatto un **sistema di equazioni esplicite alle differenze**

$$\begin{cases} x_1(r_1) &= p_1, \\ &\vdots \\ x_n(r_n) &= p_n. \end{cases}$$

dove $p_1, \ldots, p_n$ sono funzioni multivariate booleane ovvero (ANF) polinomi a coefficienti 0,1 con monomi squarefree.

- Il cifrario corrispondente è dunque l'insieme delle soluzioni $f = (f_1, \ldots, f_n) : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$ di tale sistema.

- Questi crittosistemi li chiameremo **cifrari a flusso polinomiali**.

- La ricorsione che definisce molti cifrari a flusso largamente utilizzati è di fatto un **sistema di equazioni esplicite alle differenze**

$$\begin{cases} x_1(r_1) &= p_1, \\ &\vdots \\ x_n(r_n) &= p_n. \end{cases}$$

  dove $p_1, \ldots, p_n$ sono funzioni multivariate booleane ovvero (ANF) polinomi a coefficienti 0,1 con monomi squarefree.

- Il cifrario corrispondente è dunque l'insieme delle soluzioni $f = (f_1, \ldots, f_n) : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$ di tale sistema.

- Questi crittosistemi li chiameremo **cifrari a flusso polinomiali**.

- La ricorsione che definisce molti cifrari a flusso largamente utilizzati è di fatto un **sistema di equazioni esplicite alle differenze**

$$\begin{cases} x_1(r_1) &= p_1, \\ &\vdots \\ x_n(r_n) &= p_n. \end{cases}$$

  dove $p_1, \ldots, p_n$ sono funzioni multivariate booleane ovvero (ANF) polinomi a coefficienti 0,1 con monomi squarefree.

- Il cifrario corrispondente è dunque l'insieme delle soluzioni $f = (f_1, \ldots, f_n) : \{0, 1, 2, \ldots\} \to \{0, 1\}^n$ di tale sistema.

- Questi crittosistemi li chiameremo **cifrari a flusso polinomiali**.

- La struttura esplicita di tali equazioni determina l'unicità della soluzione una volta assegnato lo stato iniziale (chiave) $x_1(0), \ldots, x_1(r_1 - 1), \ldots, x_n(0), \ldots, x_n(r_n - 1)$.
- Alcuni di questi cifrari risultano pure invertibili, cioè è possibile ottenere un qualsiasi stato passato a partire da uno stato presente.
- Questo costituisce una possibile vulnerabilità del cifrario in quanto invece che attaccare direttamente la chiave, possiamo attaccare un qualsiasi stato. Ricordiamo che il keystream è fornito solo dopo un certo numero di cicli.
- D'altra parte, l'invertibilità può essere utilizzata come una risorsa per trasformare un cifrario a flusso in uno simmetrico.
- Necessità di studiare tutte le proprietà dei cifrari a flusso polinomiali al fine di analizzarne la sicurezza e le potenzialità.

- La struttura esplicita di tali equazioni determina l'unicità della soluzione una volta assegnato lo stato iniziale (chiave) $x_1(0), \ldots, x_1(r_1 - 1), \ldots, x_n(0), \ldots, x_n(r_n - 1)$.
- Alcuni di questi cifrari risultano pure invertibili, cioè è possibile ottenere un qualsiasi stato passato a partire da uno stato presente.
- Questo costituisce una possibile vulnerabilità del cifrario in quanto invece che attaccare direttamente la chiave, possiamo attaccare un qualsiasi stato. Ricordiamo che il keystream è fornito solo dopo un certo numero di cicli.
- D'altra parte, l'invertibilità può essere utilizzata come una risorsa per trasformare un cifrario a flusso in uno simmetrico.
- Necessità di studiare tutte le proprietà dei cifrari a flusso polinomiali al fine di analizzarne la sicurezza e le potenzialità.

- La struttura esplicita di tali equazioni determina l'unicità della soluzione una volta assegnato lo stato iniziale (chiave) $x_1(0), \ldots, x_1(r_1 - 1), \ldots, x_n(0), \ldots, x_n(r_n - 1)$.
- Alcuni di questi cifrari risultano pure invertibili, cioè è possibile ottenere un qualsiasi stato passato a partire da uno stato presente.
- Questo costituisce una possibile vulnerabilità del cifrario in quanto invece che attaccare direttamente la chiave, possiamo attaccare un qualsiasi stato. Ricordiamo che il keystream è fornito solo dopo un certo numero di cicli.
- D'altra parte, l'invertibilità può essere utilizzata come una risorsa per trasformare un cifrario a flusso in uno simmetrico.
- Necessità di studiare tutte le proprietà dei cifrari a flusso polinomiali al fine di analizzarne la sicurezza e le potenzialità.

- La struttura esplicita di tali equazioni determina l'unicità della soluzione una volta assegnato lo stato iniziale (chiave) $x_1(0), \ldots, x_1(r_1 - 1), \ldots, x_n(0), \ldots, x_n(r_n - 1)$.
- Alcuni di questi cifrari risultano pure invertibili, cioè è possibile ottenere un qualsiasi stato passato a partire da uno stato presente.
- Questo costituisce una possibile vulnerabilità del cifrario in quanto invece che attaccare direttamente la chiave, possiamo attaccare un qualsiasi stato. Ricordiamo che il keystream è fornito solo dopo un certo numero di cicli.
- D'altra parte, l'invertibilità può essere utilizzata come una risorsa per trasformare un cifrario a flusso in uno simmetrico.
- Necessità di studiare tutte le proprietà dei cifrari a flusso polinomiali al fine di analizzarne la sicurezza e le potenzialità.

- La struttura esplicita di tali equazioni determina l'unicità della soluzione una volta assegnato lo stato iniziale (chiave) $x_1(0), \ldots, x_1(r_1 - 1), \ldots, x_n(0), \ldots, x_n(r_n - 1)$.
- Alcuni di questi cifrari risultano pure invertibili, cioè è possibile ottenere un qualsiasi stato passato a partire da uno stato presente.
- Questo costituisce una possibile vulnerabilità del cifrario in quanto invece che attaccare direttamente la chiave, possiamo attaccare un qualsiasi stato. Ricordiamo che il keystream è fornito solo dopo un certo numero di cicli.
- D'altra parte, l'invertibilità può essere utilizzata come una risorsa per trasformare un cifrario a flusso in uno simmetrico.
- Necessità di studiare tutte le proprietà dei cifrari a flusso polinomiali al fine di analizzarne la sicurezza e le potenzialità.

- Due cifrari a flusso polinomiali molto utilizzati sono: Trivium e Keeloq.

- **Trivium** è un cifrario sviluppato in Europa (Bart Preneel, Belgio) nell'ambito del progetto eSTREAM.
- È uno dei tre vincitori della call per lo sviluppo di cifrari a flusso efficienti in hardware.
- Produce un keystream di periodo almeno 2^{64} mediante una chiave ed un IV di 80 bit.
- Trivium è stato pubblicato nel 2003. Nonostante la sua struttura estremamente semplice, nessuno attacco definitivo è stato portato a Trivium fino ad oggi.

- Due cifrari a flusso polinomiali molto utilizzati sono: Trivium e Keeloq.
- **Trivium** è un cifrario sviluppato in Europa (Bart Preneel, Belgio) nell'ambito del progetto eSTREAM.
- È uno dei tre vincitori della call per lo sviluppo di cifrari a flusso efficienti in hardware.
- Produce un keystream di periodo almeno 2^{64} mediante una chiave ed un IV di 80 bit.
- Trivium è stato pubblicato nel 2003. Nonostante la sua struttura estremamente semplice, nessuno attacco definitivo è stato portato a Trivium fino ad oggi.

- Due cifrari a flusso polinomiali molto utilizzati sono: Trivium e Keeloq.
- **Trivium** è un cifrario sviluppato in Europa (Bart Preneel, Belgio) nell'ambito del progetto eSTREAM.
- È uno dei tre vincitori della call per lo sviluppo di cifrari a flusso efficienti in hardware.
- Produce un keystream di periodo almeno 2^{64} mediante una chiave ed un IV di 80 bit.
- Trivium è stato pubblicato nel 2003. Nonostante la sua struttura estremamente semplice, nessuno attacco definitivo è stato portato a Trivium fino ad oggi.

- Due cifrari a flusso polinomiali molto utilizzati sono: Trivium e Keeloq.
- **Trivium** è un cifrario sviluppato in Europa (Bart Preneel, Belgio) nell'ambito del progetto eSTREAM.
- È uno dei tre vincitori della call per lo sviluppo di cifrari a flusso efficienti in hardware.
- Produce un keystream di periodo almeno 2^{64} mediante una chiave ed un IV di 80 bit.
- Trivium è stato pubblicato nel 2003. Nonostante la sua struttura estremamente semplice, nessuno attacco definitivo è stato portato a Trivium fino ad oggi.

- Due cifrari a flusso polinomiali molto utilizzati sono: Trivium e Keeloq.
- **Trivium** è un cifrario sviluppato in Europa (Bart Preneel, Belgio) nell'ambito del progetto eSTREAM.
- È uno dei tre vincitori della call per lo sviluppo di cifrari a flusso efficienti in hardware.
- Produce un keystream di periodo almeno 2^{64} mediante una chiave ed un IV di 80 bit.
- Trivium è stato pubblicato nel 2003. Nonostante la sua struttura estremamente semplice, nessuno attacco definitivo è stato portato a Trivium fino ad oggi.

Trivium:

$$\begin{cases} x(93) &= z(45) + x(24) + z(0) + z(1)z(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ z(111) &= z(24) + y(15) + y(0) + y(1)y(2), \\ k(46) &= x(27) + x(0) + y(15) + y(0) + z(45) + z(0). \end{cases}$$

- Lo stato (registro) del sistema è costituito da $93 + 84 + 111 = 288$ bit, precisamente $x(0+i), \ldots, x(92+i), y(0+i), \ldots, x(83+i)$, $z(0+i), \ldots, z(110+i)$.
- L'evoluzione dello stato è definita dalle prime 3 equazioni quadratiche alle differenze.
- La quarta equazione lineare (indipendente) serve per produrre il keystream che viene fornito dopo $4 \cdot 288 = 1152$ clock.
- La chiave ed il IV costituiscono $80 + 80 = 160$ bit dello stato iniziale. I restanti bit dello stato iniziale sono prefissati.

Trivium:

$$\begin{cases} x(93) &= z(45) + x(24) + z(0) + z(1)z(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ z(111) &= z(24) + y(15) + y(0) + y(1)y(2), \\ k(46) &= x(27) + x(0) + y(15) + y(0) + z(45) + z(0). \end{cases}$$

- Lo stato (registro) del sistema è costituito da $93 + 84 + 111 = 288$ bit, precisamente $x(0+i), \ldots, x(92+i), y(0+i), \ldots, x(83+i)$, $z(0+i), \ldots, z(110+i)$.
- L'evoluzione dello stato è definita dalle prime 3 equazioni quadratiche alle differenze.
- La quarta equazione lineare (indipendente) serve per produrre il keystream che viene fornito dopo $4 \cdot 288 = 1152$ clock.
- La chiave ed il IV costituiscono $80 + 80 = 160$ bit dello stato iniziale. I restanti bit dello stato iniziale sono prefissati.

Trivium:

$$\begin{cases} x(93) &= z(45) + x(24) + z(0) + z(1)z(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ z(111) &= z(24) + y(15) + y(0) + y(1)y(2), \\ k(46) &= x(27) + x(0) + y(15) + y(0) + z(45) + z(0). \end{cases}$$

- Lo stato (registro) del sistema è costituito da $93 + 84 + 111 = 288$ bit, precisamente $x(0+i), \ldots, x(92+i), y(0+i), \ldots, x(83+i)$, $z(0+i), \ldots, z(110+i)$.
- L'evoluzione dello stato è definita dalle prime 3 equazioni quadratiche alle differenze.
- La quarta equazione lineare (indipendente) serve per produrre il keystream che viene fornito dopo $4 \cdot 288 = 1152$ clock.
- La chiave ed il IV costituiscono $80 + 80 = 160$ bit dello stato iniziale. I restanti bit dello stato iniziale sono prefissati.

Trivium:

$$\begin{cases} x(93) &= z(45) + x(24) + z(0) + z(1)z(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ z(111) &= z(24) + y(15) + y(0) + y(1)y(2), \\ k(46) &= x(27) + x(0) + y(15) + y(0) + z(45) + z(0). \end{cases}$$

- Lo stato (registro) del sistema è costituito da $93 + 84 + 111 = 288$ bit, precisamente $x(0+i), \ldots, x(92+i), y(0+i), \ldots, x(83+i)$, $z(0+i), \ldots, z(110+i)$.
- L'evoluzione dello stato è definita dalle prime 3 equazioni quadratiche alle differenze.
- La quarta equazione lineare (indipendente) serve per produrre il keystream che viene fornito dopo $4 \cdot 288 = 1152$ clock.
- La chiave ed il IV costituiscono $80 + 80 = 160$ bit dello stato iniziale. I restanti bit dello stato iniziale sono prefissati.

Trivium:

$$\begin{cases} x(93) &= z(45) + x(24) + z(0) + z(1)z(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ z(111) &= z(24) + y(15) + y(0) + y(1)y(2), \\ k(46) &= x(27) + x(0) + y(15) + y(0) + z(45) + z(0). \end{cases}$$

- Lo stato (registro) del sistema è costituito da $93 + 84 + 111 = 288$ bit, precisamente $x(0+i), \ldots, x(92+i), y(0+i), \ldots, x(83+i)$, $z(0+i), \ldots, z(110+i)$.
- L'evoluzione dello stato è definita dalle prime 3 equazioni quadratiche alle differenze.
- La quarta equazione lineare (indipendente) serve per produrre il keystream che viene fornito dopo $4 \cdot 288 = 1152$ clock.
- La chiave ed il IV costituiscono $80 + 80 = 160$ bit dello stato iniziale. I restanti bit dello stato iniziale sono prefissati.

- Una variante molto studiata in crittoanalisi è

Bivium:

$$\begin{cases} x(93) &= x(24) + y(15) + y(0) + y(1)y(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ k(28) &= x(27) + x(0) + y(15) + y(0). \end{cases}$$

- Lo stato è costituito da $93 + 84 = 177$ bits ed il keystream va in output dopo $4 \cdot 177 = 708$ clock.
- Osserviamo che entrambi Bivium e Trivium sono invertibili nel tempo e quindi è possibile attaccare lo stato corrispondente all'inizio del keystream piuttosto che quello iniziale (chiave).

- Una variante molto studiata in crittoanalisi è

Bivium:

$$\begin{cases} x(93) &= x(24) + y(15) + y(0) + y(1)y(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ k(28) &= x(27) + x(0) + y(15) + y(0). \end{cases}$$

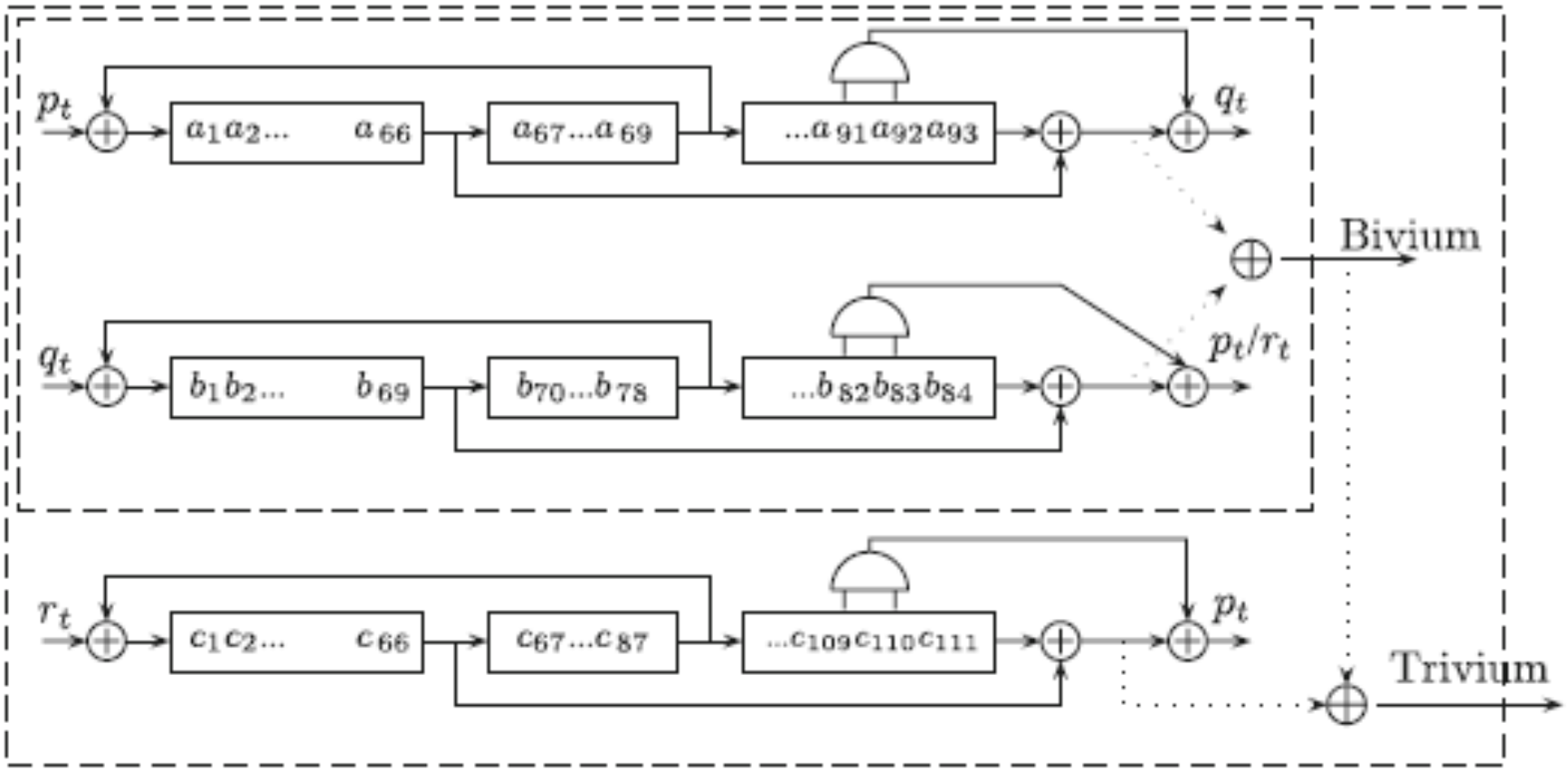
- Lo stato è costituito da $93 + 84 = 177$ bits ed il keystream va in output dopo $4 \cdot 177 = 708$ clock.
- Osserviamo che entrambi Bivium e Trivium sono invertibili nel tempo e quindi è possibile attaccare lo stato corrispondente all'inizio del keystream piuttosto che quello iniziale (chiave).

- Una variante molto studiata in crittoanalisi è

Bivium:

$$\begin{cases} x(93) &= x(24) + y(15) + y(0) + y(1)y(2), \\ y(84) &= x(27) + y(6) + x(0) + x(1)x(2), \\ k(28) &= x(27) + x(0) + y(15) + y(0). \end{cases}$$

- Lo stato è costituito da $93 + 84 = 177$ bits ed il keystream va in output dopo $4 \cdot 177 = 708$ clock.
- Osserviamo che entrambi Bivium e Trivium sono invertibili nel tempo e quindi è possibile attaccare lo stato corrispondente all'inizio del keystream piuttosto che quello iniziale (chiave).


p_t
$a_1 a_2 \ldots$ a_{66}
$a_{67} \ldots a_{69}$
$\ldots a_{91} a_{92} a_{93}$
q_t
Bivium
q_t
$b_1 b_2 \ldots$ b_{69}
$b_{70} \ldots b_{78}$
$\ldots b_{82} b_{83} b_{84}$
p_t / r_t
r_t
$c_1 c_2 \ldots$ c_{66}
$c_{67} \ldots c_{87}$
$\ldots c_{109} c_{110} c_{111}$
p_t
Trivium

- Un cifrario a flusso invertibile può essere utilizzato pure come cifrario simmetrico.
- È sufficiente considerare una parte dello stato iniziale come plaintext e la restante parte come chiave.
- Dopo un certo numero prefissato di cicli, il ciphertext è costituito da una parte dello stato finale.
- L'invertibilità del corrispondente sistema alle differenze garantisce la decifrazione.
- Un esempio fondamentale di questo utilizzo è il cifrario **Keeloq**. Ampiamente utilizzato nei transponder RFID per immobilizzatori auto, aperture porte, etc.
- Venduto il brevetto nel 1995 per 10^7 USD, è stato crittoanalizzato in modo critico nel 2008.

- Un cifrario a flusso invertibile può essere utilizzato pure come cifrario simmetrico.
- È sufficiente considerare una parte dello stato iniziale come plaintext e la restante parte come chiave.
- Dopo un certo numero prefissato di cicli, il ciphertext è costituito da una parte dello stato finale.
- L'invertibilità del corrispondente sistema alle differenze garantisce la decifrazione.
- Un esempio fondamentale di questo utilizzo è il cifrario **Keeloq**. Ampiamente utilizzato nei transponder RFID per immobilizzatori auto, aperture porte, etc.
- Venduto il brevetto nel 1995 per 10^7 USD, è stato crittoanalizzato in modo critico nel 2008.

- Un cifrario a flusso invertibile può essere utilizzato pure come cifrario simmetrico.
- È sufficiente considerare una parte dello stato iniziale come plaintext e la restante parte come chiave.
- Dopo un certo numero prefissato di cicli, il ciphertext è costituito da una parte dello stato finale.
- L'invertibilità del corrispondente sistema alle differenze garantisce la decifrazione.
- Un esempio fondamentale di questo utilizzo è il cifrario **Keeloq**. Ampiamente utilizzato nei transponder RFID per immobilizzatori auto, aperture porte, etc.
- Venduto il brevetto nel 1995 per 10^7 USD, è stato crittoanalizzato in modo critico nel 2008.

- Un cifrario a flusso invertibile può essere utilizzato pure come cifrario simmetrico.
- È sufficiente considerare una parte dello stato iniziale come plaintext e la restante parte come chiave.
- Dopo un certo numero prefissato di cicli, il ciphertext è costituito da una parte dello stato finale.
- L'invertibilità del corrispondente sistema alle differenze garantisce la decifrazione.
- Un esempio fondamentale di questo utilizzo è il cifrario **Keeloq**. Ampiamente utilizzato nei transponder RFID per immobilizzatori auto, aperture porte, etc.
- Venduto il brevetto nel 1995 per 10^7 USD, è stato crittoanalizzato in modo critico nel 2008.

- Un cifrario a flusso invertibile può essere utilizzato pure come cifrario simmetrico.
- È sufficiente considerare una parte dello stato iniziale come plaintext e la restante parte come chiave.
- Dopo un certo numero prefissato di cicli, il ciphertext è costituito da una parte dello stato finale.
- L'invertibilità del corrispondente sistema alle differenze garantisce la decifrazione.
- Un esempio fondamentale di questo utilizzo è il cifrario **Keeloq**. Ampiamente utilizzato nei transponder RFID per immobilizzatori auto, aperture porte, etc.
- Venduto il brevetto nel 1995 per 10^7 USD, è stato crittoanalizzato in modo critico nel 2008.

- Un cifrario a flusso invertibile può essere utilizzato pure come cifrario simmetrico.
- È sufficiente considerare una parte dello stato iniziale come plaintext e la restante parte come chiave.
- Dopo un certo numero prefissato di cicli, il ciphertext è costituito da una parte dello stato finale.
- L'invertibilità del corrispondente sistema alle differenze garantisce la decifrazione.
- Un esempio fondamentale di questo utilizzo è il cifrario **Keeloq**. Ampiamente utilizzato nei transponder RFID per immobilizzatori auto, aperture porte, etc.
- Venduto il brevetto nel 1995 per 10^7 USD, è stato crittoanalizzato in modo critico nel 2008.

Keeloq:

$$\begin{cases} x(32) = k(0) + x(0) + x(16) + x(9) + x(1) + x(31)x(20) \\ \qquad + x(31)x(1) + x(26)x(20) + x(26)x(1) + x(20)x(9) \\ \qquad + x(9)x(1) + x(31)x(9)x(1) + x(31)x(20)x(1) \\ \qquad + x(31)x(26)x(9) + x(31)x(26)x(20), \\ k(64) = k(0). \end{cases}$$

- Lo stato iniziale del sistema alle differenze è costituito da $32 + 64$ bit dove i primi 32 bit $x(0), \ldots, x(31)$ sono considerati plaintext ed i seguenti 64 bit $k(0), \ldots, k(63)$ sono la chiave del cifrario simmetrico.
- L'equazione lineare (indipendente) $k(64) = k(0)$ determina che l'evoluzione della chiave è semplicemente una permutazione ciclica dei suoi bit.
- Dopo $8 \cdot 64 + 16 = 528$ clock, il ciphertext è costituito dai primi 32 bit dello stato finale. La decifrazione di ottiene dal sistema alle differenze inverso.

Keeloq:

$$\begin{cases} x(32) = k(0) + x(0) + x(16) + x(9) + x(1) + x(31)x(20) \\ \qquad + x(31)x(1) + x(26)x(20) + x(26)x(1) + x(20)x(9) \\ \qquad + x(9)x(1) + x(31)x(9)x(1) + x(31)x(20)x(1) \\ \qquad + x(31)x(26)x(9) + x(31)x(26)x(20), \\ k(64) = k(0). \end{cases}$$

- Lo stato iniziale del sistema alle differenze è costituito da $32 + 64$ bit dove i primi 32 bit $x(0), \ldots, x(31)$ sono considerati plaintext ed i seguenti 64 bit $k(0), \ldots, k(63)$ sono la chiave del cifrario simmetrico.
- L'equazione lineare (indipendente) $k(64) = k(0)$ determina che l'evoluzione della chiave è semplicemente una permutazione ciclica dei suoi bit.
- Dopo $8 \cdot 64 + 16 = 528$ clock, il ciphertext è costituito dai primi 32 bit dello stato finale. La decifrazione di ottiene dal sistema alle differenze inverso.

Keeloq:

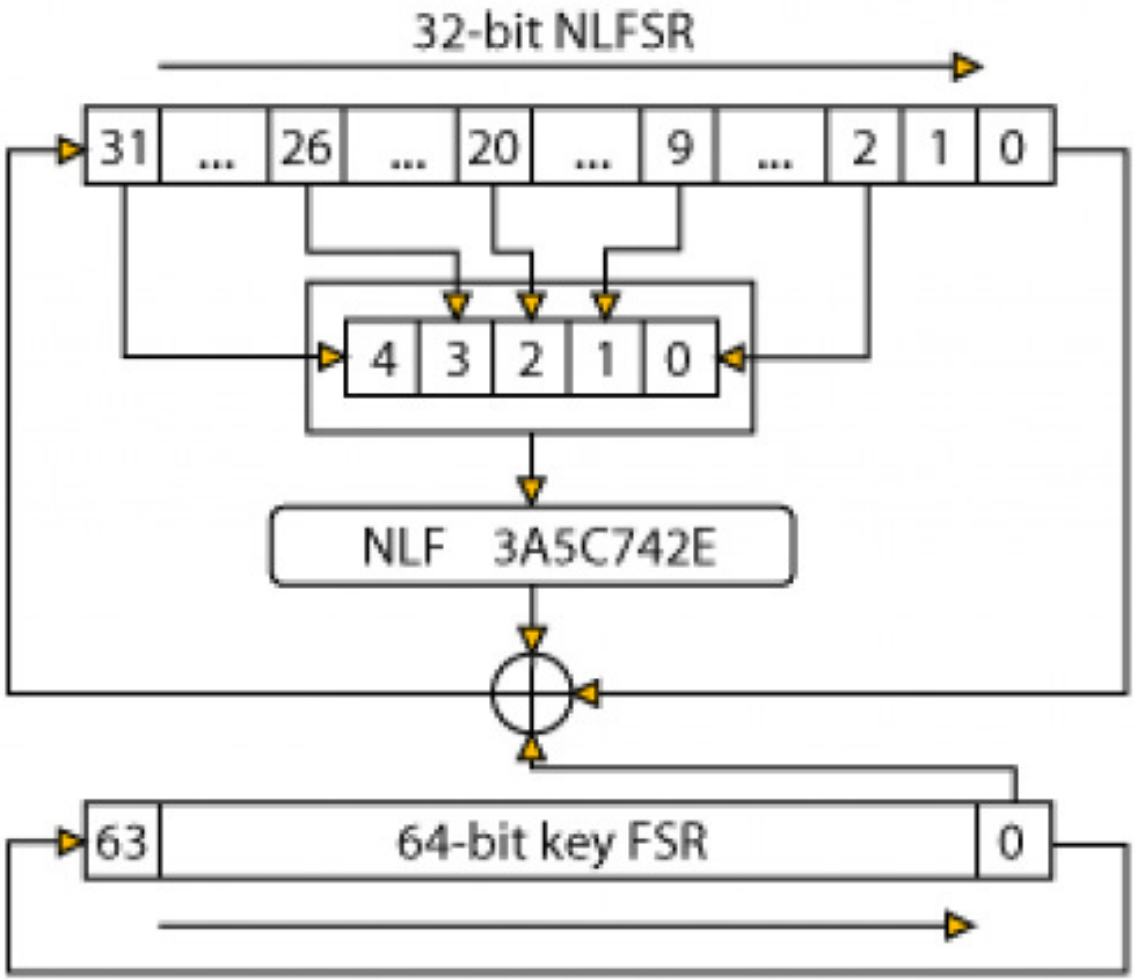
$$\begin{cases} x(32) = k(0) + x(0) + x(16) + x(9) + x(1) + x(31)x(20) \\ \qquad + x(31)x(1) + x(26)x(20) + x(26)x(1) + x(20)x(9) \\ \qquad + x(9)x(1) + x(31)x(9)x(1) + x(31)x(20)x(1) \\ \qquad + x(31)x(26)x(9) + x(31)x(26)x(20), \\ k(64) = k(0). \end{cases}$$

- Lo stato iniziale del sistema alle differenze è costituito da $32 + 64$ bit dove i primi 32 bit $x(0), \ldots, x(31)$ sono considerati plaintext ed i seguenti 64 bit $k(0), \ldots, k(63)$ sono la chiave del cifrario simmetrico.
- L'equazione lineare (indipendente) $k(64) = k(0)$ determina che l'evoluzione della chiave è semplicemente una permutazione ciclica dei suoi bit.
- Dopo $8 \cdot 64 + 16 = 528$ clock, il ciphertext è costituito dai primi 32 bit dello stato finale. La decifrazione di ottiene dal sistema alle differenze inverso.

Keeloq:

$$\begin{cases} x(32) = k(0) + x(0) + x(16) + x(9) + x(1) + x(31)x(20) \\ \qquad + x(31)x(1) + x(26)x(20) + x(26)x(1) + x(20)x(9) \\ \qquad + x(9)x(1) + x(31)x(9)x(1) + x(31)x(20)x(1) \\ \qquad + x(31)x(26)x(9) + x(31)x(26)x(20), \\ k(64) = k(0). \end{cases}$$

- Lo stato iniziale del sistema alle differenze è costituito da $32 + 64$ bit dove i primi 32 bit $x(0), \ldots, x(31)$ sono considerati plaintext ed i seguenti 64 bit $k(0), \ldots, k(63)$ sono la chiave del cifrario simmetrico.
- L'equazione lineare (indipendente) $k(64) = k(0)$ determina che l'evoluzione della chiave è semplicemente una permutazione ciclica dei suoi bit.
- Dopo $8 \cdot 64 + 16 = 528$ clock, il ciphertext è costituito dai primi 32 bit dello stato finale. La decifrazione di ottiene dal sistema alle differenze inverso.


32-bit NLFSR
31
...
26
...
20
...
9
...
2
1
0
4
3
2
1
0
NLF 3A5C742E
63
64-bit key FSR
0

- Gli attacchi di tipo algebrico ad un cifrario a flusso polinomiale

$$\begin{cases} x_1(r_1) &= p_1, \\ &\vdots \\ x_n(r_n) &= p_n. \end{cases}$$

consistono nell'assegnare la variabile corrispondente al keystream per un certo numero di cicli. Questo in quanto si assume che sia noto un numero sufficiente di coppie plaintext e ciphertext e quindi il corrispondente keystream.

- Se il numero di clock è sufficiente grande, esisterà una unica soluzione per il sistema algebrico ottenuto (evoluzione dello stato compatibile con il keystream).

- Gli attacchi di tipo algebrico ad un cifrario a flusso polinomiale

$$\begin{cases} x_1(r_1) &= p_1, \\ &\vdots \\ x_n(r_n) &= p_n. \end{cases}$$

consistono nell'assegnare la variabile corrispondente al keystream per un certo numero di cicli. Questo in quanto si assume che sia noto un numero sufficiente di coppie plaintext e ciphertext e quindi il corrispondente keystream.

- Se il numero di clock è sufficiente grande, esisterà una unica soluzione per il sistema algebrico ottenuto (evoluzione dello stato compatibile con il keystream).

- È possibile ottenere le equazioni soddisfatte dalle sole variabili dello stato iniziale (chiave) eliminando le variabili corrispondenti a tutti gli altri stati mediante le equazioni del cifrario.
- Si ottiene dunque un sistema algebrico che ammette come unica soluzione la chiave.
- Questo sistema può essere risolto con algoritmi di calcolo simbolico come le basi di Gröbner oppure con analizzatori di soddisfacibilità booleana (SAT solvers). L'efficacia dell'uno o dell'altro metodo dipende dal cifrario.
- Nonostante una certa "cattiva reputazione" di metodo inefficiente, le basi di Gröbner si dimostrano invece un risolutore potente quando se ne conoscano le numerose varianti e opzioni di applicazione.

- È possibile ottenere le equazioni soddisfatte dalle sole variabili dello stato iniziale (chiave) eliminando le variabili corrispondenti a tutti gli altri stati mediante le equazioni del cifrario.
- Si ottiene dunque un sistema algebrico che ammette come unica soluzione la chiave.
- Questo sistema può essere risolto con algoritmi di calcolo simbolico come le basi di Gröbner oppure con analizzatori di soddisfacibilità booleana (SAT solvers). L'efficacia dell'uno o dell'altro metodo dipende dal cifrario.
- Nonostante una certa "cattiva reputazione" di metodo inefficiente, le basi di Gröbner si dimostrano invece un risolutore potente quando se ne conoscano le numerose varianti e opzioni di applicazione.

- È possibile ottenere le equazioni soddisfatte dalle sole variabili dello stato iniziale (chiave) eliminando le variabili corrispondenti a tutti gli altri stati mediante le equazioni del cifrario.
- Si ottiene dunque un sistema algebrico che ammette come unica soluzione la chiave.
- Questo sistema può essere risolto con algoritmi di calcolo simbolico come le basi di Gröbner oppure con analizzatori di soddisfacibilità booleana (SAT solvers). L'efficacia dell'uno o dell'altro metodo dipende dal cifrario.
- Nonostante una certa "cattiva reputazione" di metodo inefficiente, le basi di Gröbner si dimostrano invece un risolutore potente quando se ne conoscano le numerose varianti e opzioni di applicazione.

- È possibile ottenere le equazioni soddisfatte dalle sole variabili dello stato iniziale (chiave) eliminando le variabili corrispondenti a tutti gli altri stati mediante le equazioni del cifrario.
- Si ottiene dunque un sistema algebrico che ammette come unica soluzione la chiave.
- Questo sistema può essere risolto con algoritmi di calcolo simbolico come le basi di Gröbner oppure con analizzatori di soddisfacibilità booleana (SAT solvers). L'efficacia dell'uno o dell'altro metodo dipende dal cifrario.
- Nonostante una certa "cattiva reputazione" di metodo inefficiente, le basi di Gröbner si dimostrano invece un risolutore potente quando se ne conoscano le numerose varianti e opzioni di applicazione.

- Per ottenere risultati effettivi con basi di Gröbner o SAT solvers, i sistemi devono essere generalmente semplificati mediante l'assegnazione esaustiva dei valori di un certo numero di variabili.
- Se le variabili assegnate sono k, il tempo di calcolo medio deve essere moltiplicato per 2^k. Una ottimizione cruciale è quindi la scelta delle variabili da assegnare.
- In collaborazione con Sharwan K. Tiwari, con queste ed altre tecniche abbiamo condotto attacchi algebrici ai cifrari Trivium/Bivium e Keeloq.
- Trivium ha confermato la sua solidità, mentre siamo stati in grado di ridurre significativamente i tempi di calcolo della chiave di Bivium e Keeloq.

- Per ottenere risultati effettivi con basi di Gröbner o SAT solvers, i sistemi devono essere generalmente semplificati mediante l'assegnazione esaustiva dei valori di un certo numero di variabili.
- Se le variabili assegnate sono k, il tempo di calcolo medio deve essere moltiplicato per 2^k. Una ottimizzione cruciale è quindi la scelta delle variabili da assegnare.
- In collaborazione con Sharwan K. Tiwari, con queste ed altre tecniche abbiamo condotto attacchi algebrici ai cifrari Trivium/Bivium e Keeloq.
- Trivium ha confermato la sua solidità, mentre siamo stati in grado di ridurre significativamente i tempi di calcolo della chiave di Bivium e Keeloq.

- Per ottenere risultati effettivi con basi di Gröbner o SAT solvers, i sistemi devono essere generalmente semplificati mediante l'assegnazione esaustiva dei valori di un certo numero di variabili.
- Se le variabili assegnate sono k, il tempo di calcolo medio deve essere moltiplicato per 2^k. Una ottimizione cruciale è quindi la scelta delle variabili da assegnare.
- In collaborazione con Sharwan K. Tiwari, con queste ed altre tecniche abbiamo condotto attacchi algebrici ai cifrari Trivium/Bivium e Keeloq.
- Trivium ha confermato la sua solidità, mentre siamo stati in grado di ridurre significativamente i tempi di calcolo della chiave di Bivium e Keeloq.

- Per ottenere risultati effettivi con basi di Gröbner o SAT solvers, i sistemi devono essere generalmente semplificati mediante l'assegnazione esaustiva dei valori di un certo numero di variabili.
- Se le variabili assegnate sono k, il tempo di calcolo medio deve essere moltiplicato per 2^k. Una ottimizione cruciale è quindi la scelta delle variabili da assegnare.
- In collaborazione con Sharwan K. Tiwari, con queste ed altre tecniche abbiamo condotto attacchi algebrici ai cifrari Trivium/Bivium e Keeloq.
- Trivium ha confermato la sua solidità, mentre siamo stati in grado di ridurre significativamente i tempi di calcolo della chiave di Bivium e Keeloq.

- Per l'attacco algebrico a Bivium si assegnano in modo esaustivo 38 variabili. Grazie ad equazioni lineari presenti nel sistema, si ottiene di fatto l'assegnazione di 60 variabili sul totale di 177 variabili di stato.
- Per la risoluzione delle equazioni, si è utilizzato un paio di implementazioni dell'algoritmo di Buchberger per le basi di Gröbner che si trovano nel sistema di calcolo simbolico SINGULAR.
- Abbiamo confrontato queste implementazioni con un paio SAT solver molto efficienti: MiniSat e CryptoMinisat.
- Lo speedup da noi ottenuto rispetto alla precedente letteratura è circa un fattore 40 che fa scendere la complessità dell'attacco a 2^{34} sec (singolo processore). Basandosi su assegnazioni indipendenti, l'attacco può essere parallelizato in modo naturale.

- Per l'attacco algebrico a Bivium si assegnano in modo esaustivo 38 variabili. Grazie ad equazioni lineari presenti nel sistema, si ottiene di fatto l'assegnazione di 60 variabili sul totale di 177 variabili di stato.
- Per la risoluzione delle equazioni, si è utilizzato un paio di implementazioni dell'algoritmo di Buchberger per le basi di Gröbner che si trovano nel sistema di calcolo simbolico SINGULAR.
- Abbiamo confrontato queste implementazioni con un paio SAT solver molto efficienti: MiniSat e CryptoMinisat.
- Lo speedup da noi ottenuto rispetto alla precedente letteratura è circa un fattore 40 che fa scendere la complessità dell'attacco a 2^{34} sec (singolo processore). Basandosi su assegnazioni indipendenti, l'attacco può essere parallelizato in modo naturale.

- Per l'attacco algebrico a Bivium si assegnano in modo esaustivo 38 variabili. Grazie ad equazioni lineari presenti nel sistema, si ottiene di fatto l'assegnazione di 60 variabili sul totale di 177 variabili di stato.
- Per la risoluzione delle equazioni, si è utilizzato un paio di implementazioni dell'algoritmo di Buchberger per le basi di Gröbner che si trovano nel sistema di calcolo simbolico SINGULAR.
- Abbiamo confrontato queste implementazioni con un paio SAT solver molto efficienti: MiniSat e CryptoMinisat.
- Lo speedup da noi ottenuto rispetto alla precedente letteratura è circa un fattore 40 che fa scendere la complessità dell'attacco a 2^{34} sec (singolo processore). Basandosi su assegnazioni indipendenti, l'attacco può essere parallelizato in modo naturale.

- Per l'attacco algebrico a Bivium si assegnano in modo esaustivo 38 variabili. Grazie ad equazioni lineari presenti nel sistema, si ottiene di fatto l'assegnazione di 60 variabili sul totale di 177 variabili di stato.
- Per la risoluzione delle equazioni, si è utilizzato un paio di implementazioni dell'algoritmo di Buchberger per le basi di Gröbner che si trovano nel sistema di calcolo simbolico SINGULAR.
- Abbiamo confrontato queste implementazioni con un paio SAT solver molto efficienti: MiniSat e CryptoMinisat.
- Lo speedup da noi ottenuto rispetto alla precedente letteratura è circa un fattore 40 che fa scendere la complessità dell'attacco a 2^{34} sec (singolo processore). Basandosi su assegnazioni indipendenti, l'attacco può essere parallelizato in modo naturale.

Assegnazioni corrette

# ks bits	slimgb	std	MiniSat	CrMiniSat
180	175 ms	338 ms	15.41 s	14.66 s
185	162 ms	332 ms	12.68 s	13.25 s
190	119 ms	336 ms	10.71 s	11.85 s
195	151 ms	418 ms	10.95 s	14.39 s

Assegnazioni errate

# ks bits	slimgb	std	MiniSat	CrMiniSat
180	173 ms	352 ms	52 s	25.52 s
185	170 ms	368 ms	34.62 s	19.61 s
190	119 ms	364 ms	37.28 s	19.72 s
195	129 ms	381 ms	37.41 s	18.85 s

- È interessante notare che le assegnazioni errate (tutte meno una) vedono prevalere in modo ancora maggiore le basi di Gröbner.
- Questo perchè in caso di non soddisfacibilità, un SAT solver deve analizzare tutto lo spazio delle soluzioni mentre una base di Gröbner deve contenere una singola equazione $1 = 0$.

GRAZIE PER L'ATTENZIONE

- È interessante notare che le assegnazioni errate (tutte meno una) vedono prevalere in modo ancora maggiore le basi di Gröbner.
- Questo perchè in caso di non soddisfacibilità, un SAT solver deve analizzare tutto lo spazio delle soluzioni mentre una base di Gröbner deve contenere una singola equazione $1 = 0$.

GRAZIE PER L'ATTENZIONE

- È interessante notare che le assegnazioni errate (tutte meno una) vedono prevalere in modo ancora maggiore le basi di Gröbner.
- Questo perchè in caso di non soddisfacibilità, un SAT solver deve analizzare tutto lo spazio delle soluzioni mentre una base di Gröbner deve contenere una singola equazione $1 = 0$.

GRAZIE PER L'ATTENZIONE