به نام خدا

جلسه هفتم :

- Message integrity
- Message Authentication Code (MAC)
- Constructing secure MACs
- CBC-MAC

• message integrity :

له یکی از مهمترین اهداف رمزنگاری ایجاد راهی برای برقرارکردن یک ارتباط امن از طریق یک کانال ناامن است.

تاکنون دیدیم چگونه می‌توان از طریق یک کانال ناامن ، ارتباطی پنهان (secret) برقرار کرد.

اما ارتباط امن فقط به معنای پنهان نگه داشتن پیام (در مقابل مهاجم مسموم کننده) نیست.

مقوله امانت و سندیت پیام (message integrity/authentication) نیز از اهمیت بالایی برخوردار است.

له توانایی احراز هویت فرستنده پیام (تضمین بر اینکه پیام در هنگام انتقال تغییری نکرده است)

مثال‌ی از کاربردهای آن : ارتباط انلاین با بانک‌ها ، کوکی‌ها (web cookies)

خرید آنلاین ، ایمیل و پیام کوتاه

* سیستم های رمزنگاری (در حالت کلی) تضمینی بر integrity پیام‌ها نمی‌دهند . در نتیجه هیچگاه نباید از آن‌ها برای احراز هویت پیام‌ها استفاده کرد. (مگر با این هدف ساخته شده باشند)

متن های رمزی تولید شده توسط سیستم های رمزنگاری پیام را به طور کامل مخفی می کنند، اما این بدان معنا نیست که مهاجم قادر به تغییر پیام نمی باشد. (تمام سیستم های رمزنگاری که تاکنون خوانده‌ایم ، integrity را تضمین نمی‌کنند)

مثال : $c = k \oplus m$ (OTP) را درنظر بگیرید- تغییر هر بیت c ، منجر به تغییر بیت متناظر در پیام می‌شود بدون آنکه امنیت سیستم زیر سوال برود.

له این حمله روی OFB-mode و CTR-mode نیز عمل می‌کند...

• کد اصالت سنجی : (Message Authentication Code (MAC))

هدف این ابزار رمزنگاری جلوگیری از تغییر پیام فرستنده توسط مهاجم ، یا صرف فرستادن پیامی جدید از طرف فرستنده توسط مهاجم است.

برای اینکار ، گیرنده و فرستنده کلید k را باهم به اشتراک می گذارند. (مانند سیستم‌های رمزمتقارن)

$MAC = (Gen, Mac, Verify)$

الگوریتم تولید کلید ، $Gen(1^n) \to k$: $|k| \geq n$ (اغلب $k \xleftarrow{R} \{0,1\}^n$)

الگوریتم تولید برچسب (tag) ، $m \in \{0,1\}^*$ ، احتمالاتی یا قطعی : $Mac_k(m) \to t$

الگوریتم قطعی تایید ، خروجی ۱ اگر t برچسب درست برای m باشد : $Verify_k(m,t) \to \{0,1\}$

شرط صحت :

$$\forall n , \forall k \leftarrow Gen(1^n) , \forall m \in \{0,1\}^* :$$

$$Verify_k(m, Mac_k(m)) = 1$$

* اگر Mac_k فقط برای پیام‌هایی با طول $l(n)$ تعریف شده باشد ، سیستم را یک کد اصالت سنجی پیام با طول ثابت برای پیام‌های به طول $l(n)$ می نامیم.

الگوریتم تایید متعارف (canonical) : اگر الگوریتم Mac قطعی باشد ، روش رایج برای تایید (Verify) کردن برچسب ، دوباره محاسبه کردن آن است. به عبارتی الگوریتم $Verify_k(m,t)$ ابتدا $\tilde{t} = Mac_k(m)$ را محاسبه کرده و سپس تساوی $t = \tilde{t}$ را چک می‌کند.

امنیت : هدف آن است که حریف مهاجم نباید قادر به تولید یک برچسب معتبر برای یک پیام جدید باشد.

برابر پیام‌های دیده‌شده توسط مهاجم نباشد.

قدرت مهاجم را مانند قبل به مهاجم‌های احتمالاتی چندجمله‌ای (PPT) محدود می‌کنیم و شکست یک کد اصالت سنجی را زمانی در نظر می‌گیریم که مهاجم پیام m و برچسب t را خروجی می‌دهد به طوری که ۱. t برچسب معتبری برای m باشد و ۲. مهاجم قبلا برچسبی برای m ندیده باشد.

(دیدن m دوم را با دادن دسترسی اوراکلی به الگوریتم $Mac_k(\cdot)$ به مهاجم مدل می‌کنیم)

- existentially unforgeable under an adaptive chosen-message attack:

  - adaptive chosen-message attack: حمله‌کننده قادر به دریافت برچسب برای پیام‌های دلخواه است.
  - existentially unforgeable: حمله‌کننده نباید قادر به جعل برچسب حتی برای هیچ پیامی باشد.

$\Pi = (Gen, Mac, Verify)$: $Mac\text{-}forge_{\mathcal{A},\Pi}(n)$ آزمایش جعل‌پذیری

۱. $k \leftarrow Gen(1^n)$

۲. $(m,t) \leftarrow \mathcal{A}^{Mac_k(\cdot)}(1^n)$ ، $Q := \{\text{all queries } \mathcal{A} \text{ asked}\}$

۳. می‌گوییم $\mathcal{A}$ موفق شده است اگر و تنها اگر ۱) $Verify_k(m,t)=1$ ۲) $m \notin Q$

در این حالت خروجی آزمایش برابر یک و در غیر اینصورت برابر صفر می‌شود.

تعریف: یک کد اصالت بخش $\Pi=(Gen,Mac,Verify)$ امنیت جعل‌ناپذیری دارد (یا existentially unforgeable under an adaptive chosen-message attack) اگر برای هر مهاجم احتمالاتی چندجمله‌ای (PPT) مانند $\mathcal{A}$، تابع ناچیز negl وجود دارد به طوری که:

$$Pr[Mac\text{-}forge_{\Pi,\mathcal{A}}(n)=1] \leq negl(n)$$

- این تعریف قوی است زیرا ۱) مهاجم می‌تواند برچسب هر پیام به دلخواه خود را ببیند و ۲) می‌گوییم مهاجم سیستم را شکسته است اگر برچسب معتبر برای یک پیام جدید ضرب دهد.
- این تعریف تضمینی در مقابل حمله بازپخش (replay attack) نمی‌کند. برای جلوگیری از این حمله اغلب از برچسب‌های زمانی (timestamps) یا شمارنده‌ها (counters) استفاده می‌شود.

- کد اصالت بخش قوی:

طبق تعریف بالا، MAC تضمین می‌کند که اگر مهاجم برچسب‌های $t_1, \ldots$ را برای پیام‌های $m_1, \ldots$ را داشته باشد، نمی‌تواند برچسب معتبر t برای پیام $m \notin \{m_1, \ldots\}$ تولید کند.

آزمایش تضمین‌یافته Mac-sforge مانند قبل تعریف می‌شود، اما اینبار مجموعه Q شامل زوج‌های (m,t) می‌باشد. مهاجم موفق می‌شود هرگاه (m,t) خروجی دهد که $Verify_k(m,t)=1$ و $(m,t) \notin Q$.

تعریف: یک کد احتمالات تجزی $\Pi=(Gen, Mac, Verify)$ دارای امنیت قوی است اگر برای هر مهاجم PPT مانند A، تابع ناچیز $negl$ موجود است به طوری که:

$$Pr[Mac\_sforge_{A,\Pi}(n)=1] \leq negl(n)$$

* اگر یک MAC امن از الگوریتم تایید رایج (canonical) استفاده کند، به صورت قوی امن است. (چرا؟)

حمله کوئری به الگوریتم تایید برچسب (verification): مهاجم را در نظر بگیرید که با گیرنده پیام تعامل دارد و با فرستادن m', t' می‌تواند درستی تساوی $Verify_k(m', t')=1$ را چک کند.

این مهاجم را می‌توان با دادن دسترسی اوراکلی به الگوریتم $Verify_k(\cdot,\cdot)$ مدل کرد.

* MACهایی که از canonical verification استفاده می‌کنند، این دسترسی تفاوتی ایجاد نمی‌کند. (چرا؟)

* MAC های قوی (دارای امنیت قوی) با دادن این دسترسی به مهاجم نیز امن باقی می‌مانند.

حمله زمانبندی (timing attack): مهاجم را در نظر بگیرید که با استفاده از تعامل با گیرنده و استفاده از آن به عنوان اوراکل الگوریتم verify، نه تنها می‌تواند اعتبار برچسب را یاد بگیرد، بلکه می‌تواند زمانی که گیرنده برای تصمیم‌گیری آن صرف می‌کند را بسنجد.

* مثالی از این حمله‌ها side-channel attack ها هستند.

فرض کنید سیستم MAC از canonical verification استفاده می‌کند. پس برای تایید اعتبار برچسب t برای پیام m، گیرنده $t':=Mac_k(m)$ را محاسبه کرده و تساوی $t=t'$ را چک می‌کند.

اگر این مقایسه با روش‌های استاندارد مقایسه، هر بایت انجام شود و به محض رسیدن به ناسازی متوقف شود، زمان عدم تایید اعتبار برچسب به اولین مکان متفاوت در t' (با t) بستگی خواهد داشت. ⟸ برای هر پیام مانند m می‌توان برچسب t جعل کرد.

مثال واقعی از حمله‌ای مشابه در Xbox 360 است.

• ساختن کد امالت سجر امن :

+ MAC برای پیام با طول ثابت :

↵ استفاده از PRF ها (احتمال حدس مقدار یک تابع تصادفی 2^{-n} است)

فرض کنید F یک تابع شبه تصادفی است. MAC برای پیام با طول n را به صورت زیر در نظر بگیرید :

$Mac_k(m)$ ← t : که $k,m \in \{0,1\}^n$ ، $t := F_k(m)$ ، (اگر $|m| \neq |k|$ آنگاه $\perp$)

$Verify_k(m,t)$ ← $0/1$: که $k,m \in \{0,1\}^n$ ، $t \in \{0,1\}^n$ ، خروجی 1 می‌دهد اگر $t = F_k(m)$

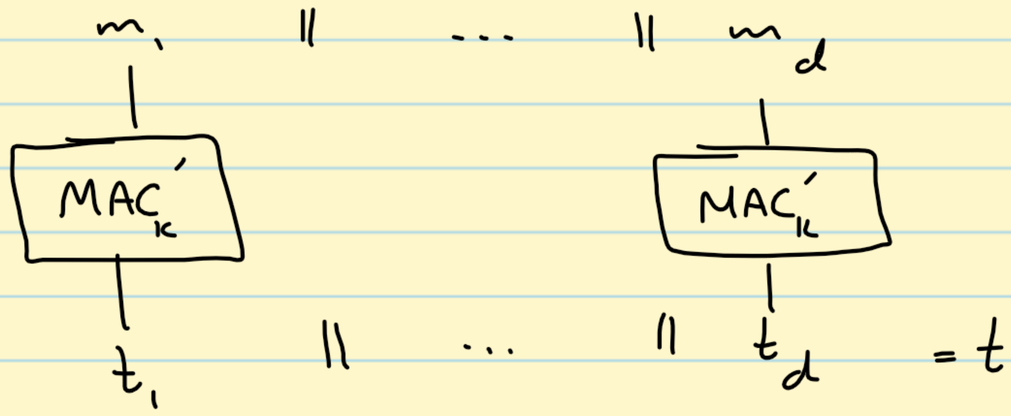
↵ اگر F یک تابع شبه تصادفی باشد، آن‌گاه این MAC برای پیام‌هایی با طول n امن است. (تمرین: اثبات کنید)

+ MAC برای پیام‌هایی با طول دلخواه :

فرض کنید $\Pi' = (Mac', Verify')$ یک MAC امن برای پیام‌هایی با طول n باشد.

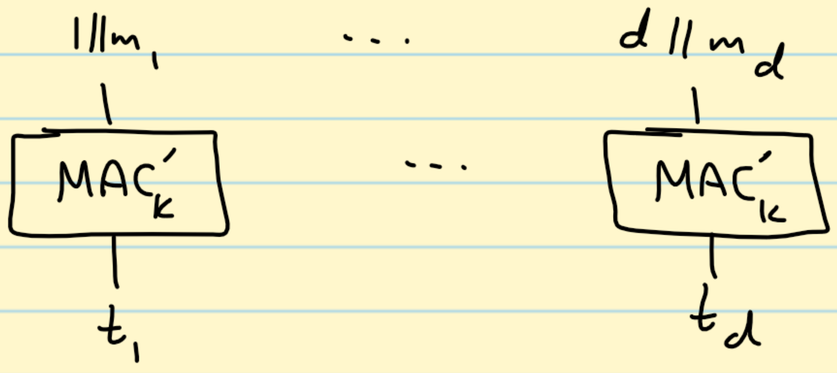
می‌خواهیم تلاش کنیم یک MAC امن برای پیام‌هایی با طول دلخواه بسازیم :

۱. ایده اول :



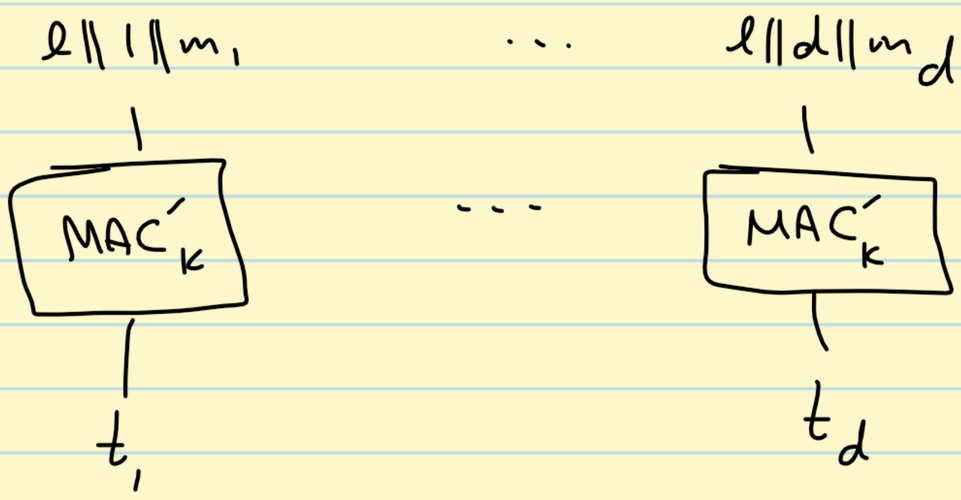
(تمرین : چرا امن نیست ؟)

۲. ایده دوم :



(تمرین : چرا امن نیست ؟)

۳. ایده سوم :

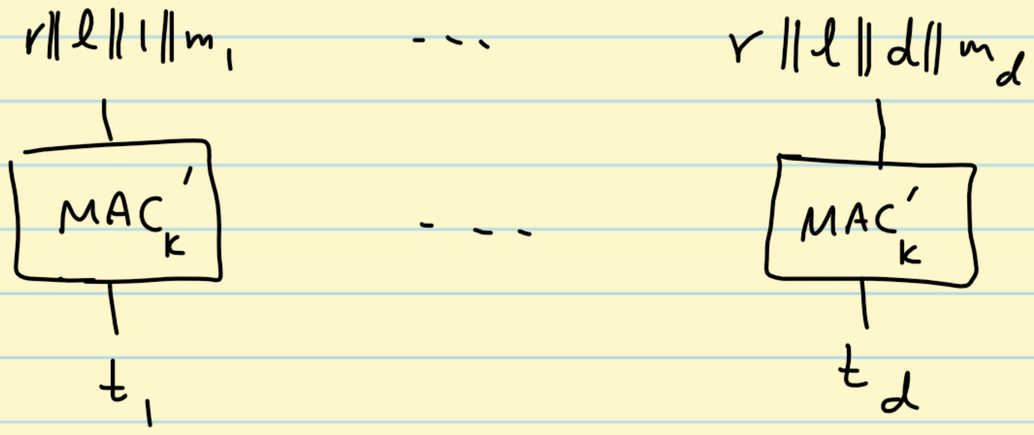


where $l = |m|$

(تمرین : چرا امن نیست ؟)

⇐ ایده نهایی : برای $k \in \{0,1\}^n$ ، $m \in \{0,1\}^*$ با طول $l < 2^{n/4}$ که $m = m_1 \| \dots \| m_d$

به طوری که $|m_1| = n/4$ ، $r \xleftarrow{\$} \{0,1\}^{n/4}$:

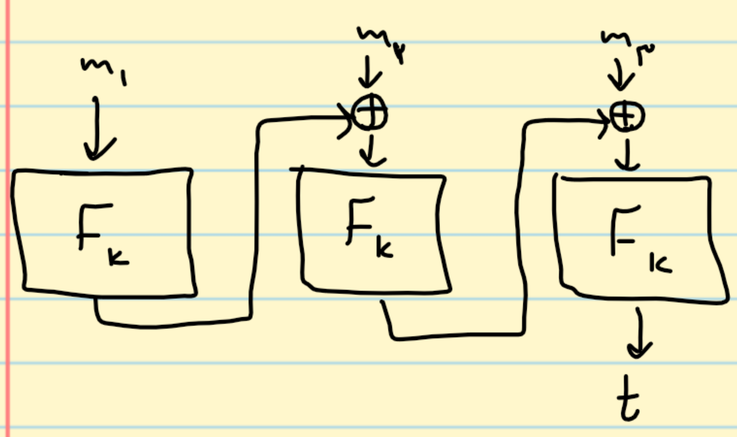


* اگر Π' یک کد اصالت بخش امن برای پیام هایی با طول n باشد، ساختار بالا یک کد اصالت بخش امن برای پیام هایی با طول دلخواه خواهد بود. (تمرین : چرا ؟)

- CBC-MAC : ساختار بالا برچسب با طول nd محاسبه می‌کند و قالب ها $4d$ بار محاسبه می‌شوند و در نتیجه ساختاری ناکارآست. در عمل CBC-MAC به عنوان یک کد اصالت بخش استاندارد استفاده می‌شود.

ساختار ساده CBC-MAC به صورت زیر است :

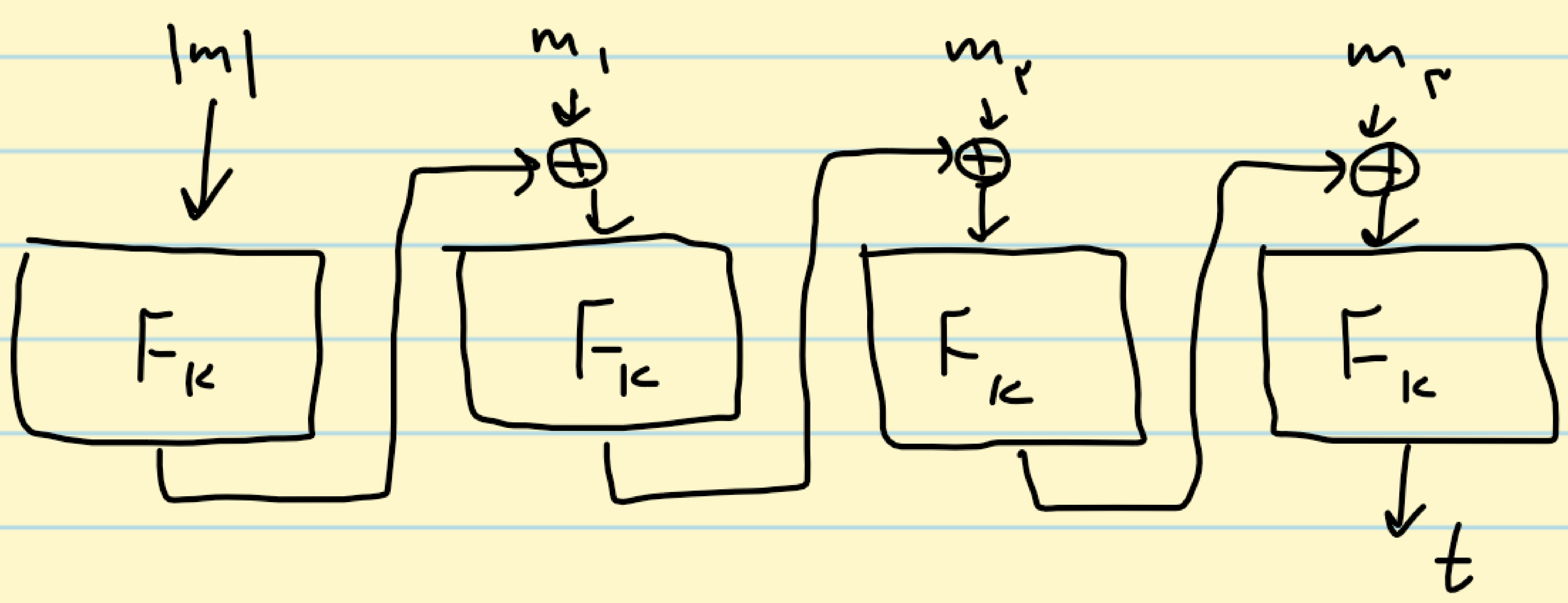
(برای پیام هایی با طول ثابت)



* اگر F یک تابع شبه تصادفی باشد، ساختار بالا یک MAC امن برای پیام‌های با طول $l(n) \cdot n$ است.

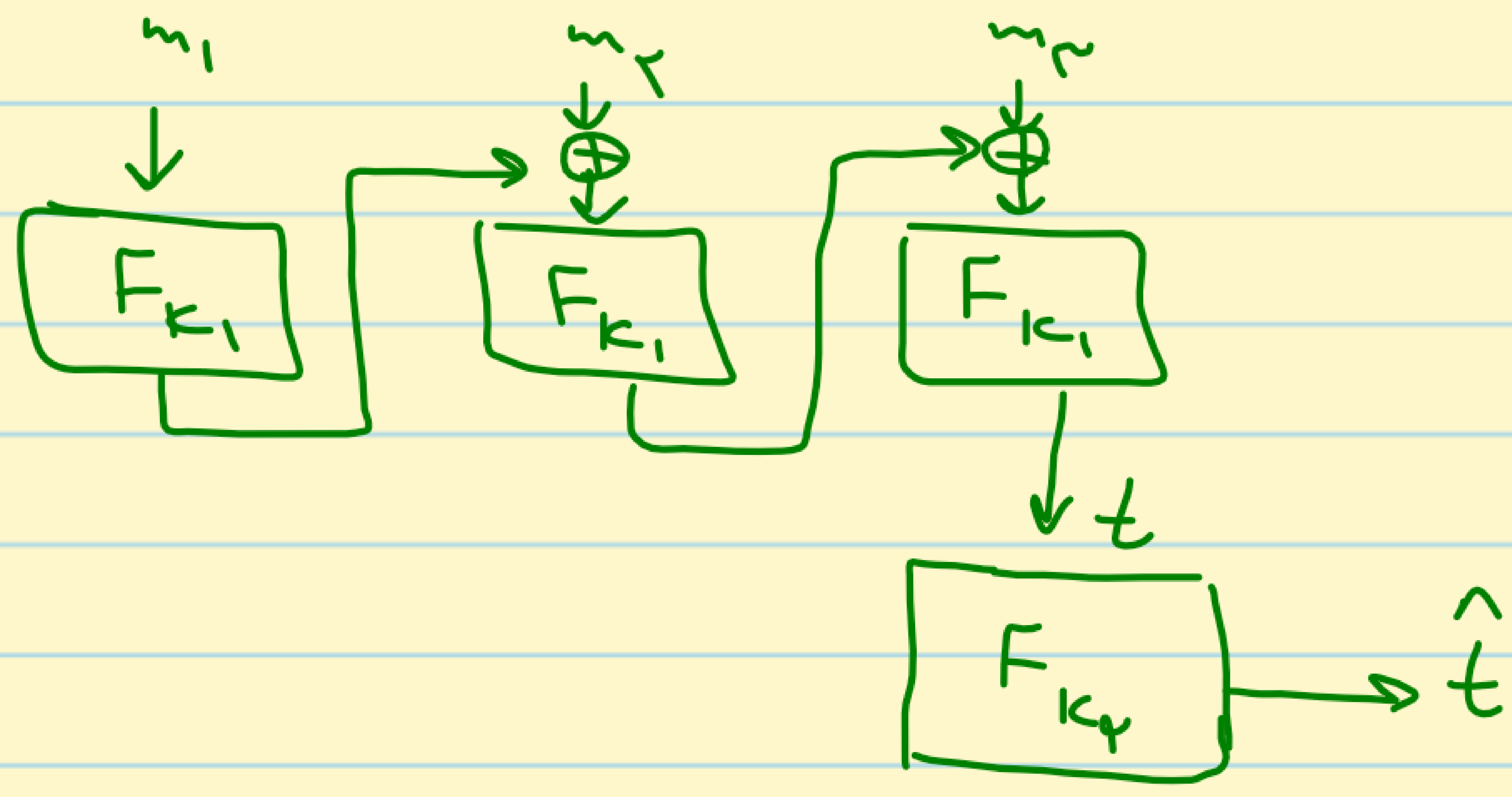
ساختار قبل زمانی امن است که طول پیام از قبل فیکس شده باشد. این ساختار کارا است و فقط به d محاسبه تابع برای پیام‌های با طول dn و تولید برچسب با طول n نیاز دارد.

* CBC-MAC امن برای پیام‌های با طول دلخواه:



* اضافه کردن |m| به آخر پیام باعث ناامن بودن CBC-MAC می‌شود. (چرا؟)

* روش دیگر تغییر CBC-MAC آن است که به جای اضافه کردن |m|، ۲ کلید مستقل $k_1, k_2 \in \{0,1\}^n$ تولید کنیم و سپس برای تولید برچسب برای پیام m به صورت زیر عمل می‌کنیم:



مزیت روش دوم آن است که نیازی به دانستن |m| نداریم و از معایب آن نیاز به وجود ۲ کلید برای F است.