دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش درس ریاضیات رمزنگاری

عنوان:

رمزگذاری جستجوپذیر متقارن پویا

استاد درس:

مهندس

نگارنده:

ز

۹۴

دی ماه ۱۳۹۴

□□□□□□□□□

چکیده

با استفاده از رمزگذاری جستجوپذیر متقارن کاربر می تواند داده های خود را به گونه ای رمز کند که قابل جستجو باشند. یکی از کاربردهای مهم این نوع رمزگذاری در رایانش ابری است که کاربر داده های خود را روی یک ابر عمومی که لزوماً مورد اعتماد او نیست ذخیره سازی می کند. یک طرح جستجوپذیر با سه ویژگی را می توان عملی دانست : امکان به روزرسانی یا اضافه و پاک کردن فایل ها(پویایی)، زمان جستجوی خطی و امنیت در برابر حمله کلیدواژه منتخب وفقی. طرح های بسیاری تا کنون برای رمزگذاری جستجوپذیر متقارن مطرح شده اند که هرکدام یک یا چندی از این ویژگی ها را برآورده می سازند. در این مقاله ابتدا به بررسی طرح های موجود و مقایسه عملکرد آن ها می پردازیم ، سپس یک طرح دارای ویژگی پویایی و امنیت اثبات پذیر در برابر حمله کلیدواژه منتخب وفقی را به تفصیل شرح می دهیم.

کلمات کلیدی: ر طرح ارائه شده در [۶] به یک پیچیدگی زمانی ثابت دست یافته است، در عین حال از به روز رسانی داده ها پشتیانی نمی کند.به این معنی که برای اضافه یا پاک کردن یک فایل باید تمامی فایل ها از اول اندیس گذاری شوند.

طرح [۳] [۶] برای هر به روزرسانی می بایست تمامی داده ها را تغییر دهیم که بسیار زمان بر و ناکارآمد خواهدبود.

-نشت اطلاعات مربوط به الگوی جستجوی [۱] کاربر را مورد توجه قرار می دهد.

تا سال ۲۰۱۲ این تنها طرح معرفی شده با دو ویژگی بالاست که امنیت آن در برابر حمله کلیدواژه منتخب وفقی [۲] اثبات شده است [۷]. این طرح در سال های بعد مبنایی برای طرح مختلف مانند [۸] قرار گرفت. قابل ذکر است که محدودیت عمده این روش بزرگ بودن اندیس رمز شده می باشد. جدول ۱ به طور خلاصه مقایسه ای از طرح های مختلف رمزنگاری متقارن و قابلیت های هر کدام را نشان می دهد:

[۱] Search pattern
[۲] adaptive chosen keyword attacks

□□□□□ ۱: مقایسه طرح های مختلف

طرح	پویایی	زمان جستجو	اندازه اندیس
SWP [۱]	-	$O(\|f\|)$	اندیس ندارد
SSE-1 [۶]	-	$O(\#f_w)$	$O(\Sigma_w \#f_w + \#W)$
SSE-2 [۶]	-	$O(\#f)$	$O(\#f.\#W)$
[۷]	√	$O(\#f_w)$	$O(\Sigma_w \#f_w + \#W)$
vLSDHJ10 [۳]	√	$O(Log\#W)$	$O(\#W.m_f)$

۱ نماد گذاری و تعریف مسئله

فرض کنید کاربر می خواهدn فایل $D_i = (M_i, W_i), i = 1, 2, ..., n$ را روی کارگزار ذخیره کند . M_i داده اصلی و W_i مجموعه ای از کلید واژه ها را نشان می دهد که به داده اصلی موجود در هر فایل افزوده شده است:

$$W_i = \{w_1, w_2, ..., w_n\}$$

W_i را افزونه فایل i ام می نامیم. هر یک از فایل ها با یک شناسه [۳] شناخته می شوند به طوری که ID_i شناسه مربوط به فایل D_i است. فایل ها باید به گونه ای روی کارگزار ذخیره گردند که محرمانگی M_i و W_i متناظر آن حفظ شود.

هدف کاربر جستجوی کلید واژه ای مانند w و یافتن M_i ای است که به ازای آن $w \in W_i$ باشد. در رمز نگاری جستجوپذیر تعریف امنیت عدم اطلاع کارگزار از هیچ یک از افزونه ها به جز افزونه فایل حاوی کلیدواژه مورد نظر است.به بیان دیگر برای حفظ امنیت هیچ اطلاعاتی به جز نتیجه جستجو نباید به کارگزار نشت کند. روشهای معمول رمزگذاری جستجوپذیر متقارن در چهار مرحله زیر عمل جستجو را انجام می دهند:

۱- keygen(s) : یک الگوریتم تولید کلید است که توسط کاربر اجرا می شود .ورودی آن یک پارامتر امنیتیsو خروجی آن کلیدKمی باشد. به طوریکه:

$$K = (k_M, k_W)$$

کلید K_M برای رمز کردن داده اصلی و کلید K_W برای رمز کردن افزونه استفاده می شود.

۲- Document-Storage(D) : فایل Dبا استفاده از کلید Kرمز شده و روی کارگزار ذخیره می شود.این تابع خود از دو زیر تابع زیر تشکیل می شود:

- **Data-Storage(M,kM) :** داده اصلیMرا با استفاده از کلید K_Mرمز کرده و خروجی E_{K_M} ، ID_iرا پس می دهد.

[۳] Document identifier

– **Metadata-Storage(W,kW) :** افزونه w را با استفاده از کلید K_W به یک نمایش جستجوپذیر S_W تبدیل میکند .

۳– **Trapdoor(w,kW) :** کاربر با استفاده از w و k_W دریچه T_w را ایجاد می کند .

۴– **Search(Tw,SW) :** کارگزار با داشتن S_W و دریچه عمل جستجو را انجام می دهد و در صورتی که $w \in W_i$ باشد خروجی ۱ را پس می دهد.

واضح است که انجام چهار مرحله بالا برای یک عمل جستجو به $O(n)$ زمان احتیاج دارد .چون کارگزار با دریافت یک T_W باید الگوریتم **Search(Tw,SW)** را روی تمامی S_W های ذخیره شده اجرا کند.

طرح های SWP [۱] و [۶] برخلاف روش بالا به جای تبدیل هر افزونه به یک نمایش جستجوپذیر هر کلیدواژه را به یک نمایش جستجو پذیر تبدیل می کنند این باعث می شود برای هر به روزرسانی [۴] مجبور باشیم تمامی داده ها را تغییر دهیم که بسیار زمان بر و ناکارآمد خواهدبود.

۲ رمزگذاری جستجوپذیر متقارن پویا [۳]

فرض کنید کاربر به دنبال فایل هایی است که کلید واژه w در آن ها اتفاق افتاده اند.برای این کار مجموعه I_W به صورت زیر تشکیل می شود:

$$I_W = \{ID_i | w \in W_i\}$$

این مجموعه شامل شناسه تمام فایل هایی است که کلیدواژه مورد جستجو در افزونه آن ها یافت شده است.ایده اصلی این طرح تبدیل هر کلید واژه w به یک نمایش جستجوپذیر S_Wاست به گونه ای که کاربر بتواند بااستفاده از دریچه T_w به فایل های حاوی w دست یابد.برای این مقصود نمایش جستجوپذیر هر کلیدواژه w به صورت زیر تشکیل می شود .

$$S_W = (f_{k_f}(w), m(I_w), R(w)) \quad (۱)$$

که در آن F_{k_f} یک تابع شبه تصادفی و k_f همان کلید خصوصی K_W یا بخشی از آن می باشد. m تابع پوشش گذار [۵] و R حاوی اطلاعاتی برای برداشتن پوشش I_w می باشد. هر بار که کاربر می خواهد کلید واژه ای مانند w را جستجو کند دریچه زیر را برای کارگزار می فرستد :

$$T_w = (f_{k_f}(w), R^{'}(w)) \quad (۲)$$

[۴] Update
[۵] Masking function

کارگزار با داشتن دریچه ابتدا تمامی نمایش های جستجوپذیر S_W را برای یافتن $F_{k_f(w)}$ جستجو میکند .هر جا که انطباقی رخ داد $m(I_w)$ متناظر با استفاده از $R(w)$ و $R^{'}(w)$ پوشش آن حذف می شود. سپس کارگزار شناسه مربوط به فایل هایی که در I_w اتفاق افتاده اند را برای کاربر می فرستد.به این ترتیب عمل جستجو به پایان می رسد. در این روش برای مخفی کردن الگوی جستجوی کاربر I_w را تا زمان انجام جستجو توسط تابع پوشش m پوشیده نگه می داریم.بسته به نوع پوشش استفاده شده این طرح را می توان به دو صورت تعاملی و غیر تعاملی استفاده نمود. طرح اول بار محاسباتی کمتری دارد در عین حال به صورت تعاملی بین کاربر و کارگزار انجام می شود.در عین حال طرح دوم به صورت غیر تعاملی عمل کرده و عمل ذخیره سازی و جستجو را با بکارگیری یک زنجیره چکیده ساز انجام می دهد.

۱.۲ طرح ۱:جستجو و ذخیره به صورت تعاملی

در این طرح نمایش جستجوپذیر هر کلیدواژه w به صورت زیر است:

$$S_W = (f_{k_f}(w), I_w \oplus G(r), E_{k_E}(w)) \quad (۳)$$

برای دست یابی به این نمایش جستجوپذیر ابتدا توابع **keygen(s)** و **Data-Storage(M,kM)** مذکور در بخش ۴ محاسبه شده ، سپس برای انجام مرحله ذخیره سازی افزونه ها الگوریتم زیر به صورت تعاملی بین کاربر و کارگزار اجرا می شود:

۱- کاربر مجموعه U_w که شامل شناسه افزونه های شامل کلیدواژه w است را به صورت زیر تشکیل داده و $f_{k_f}(w)$ را به کارگزار ارسال می کند :

۲- کارگزار اولین مولفه ی نمایش های جستجوپذیر را برای یافتن $f_{k_f}(w)$ جستجو می کند و $E_{k_E(r)}$ (مولفه سوم نمایش جستجوپذیر متناظر) را برای کاربر ارسال می کند.

۳- کاربر با داشتن $E_{k_E}(r)$ و رمزگشایی آن به صورت $r = E_{k_E}^{-1}(E_{K_E}(r))$ ، r را بدست می آورد .سپس با انتخاب $r^{'}$ به صورت تصادفی C را به صورت زیر محاسبه و به کارگزار ارسال می کند:

$$C = (f_{k_f}(w), U_w \oplus G(r) \oplus G^{'}(r), E_{k_E}(r^{'})) \quad (۴)$$

۴- نمایش جستجوپذیر به روزرسانی شده به صورت زیر خواهد بود:

$$C = (f_{k_f}(w), U_w \oplus I_w \oplus G^{(}r^{\prime}), E_{k_E}(r^{'})) \quad (۵)$$

به عبارت دیگر لیست به روز رسانی شده اندیس های به دست آمده از جستجو برابر با $I^{'}(w) = U_w \oplus I_w$ است.

حال نوبت به تولید دریچه توسط کاربر می رسد. به این منظور کاربر با داشتن کلیدخصوصی $K_W = (K_f, K_W)$ دریچه $T_w = f_{k_f}(w)$ را تولید کرده و برای کارگزار می فرستد. این دریچه در واقع مجوز جستجو است که توسط کاربر به کارگزار داده می شود.در آخر الگوریتم جستجو طی دو مرحله زیر به صورت تعاملی بین کاربر و کارگزار انجام می شود:

۱- کارگزار که دریچه T_w را در اختیار دارد اولین مولفه نمایش های جستجوپذیر S_w را برای یافتن T_w بررسی میکند و در صورد وقوع انطباق مولفه سوم نمایش جستجوپذیر متناظر (S_3) را برای کاربر ارسال میکند.

۲- کاربر $E_{k_E}^{-1}(S_3)$ را برای کارگزار می فرستد تا کارگزار طبق رابطه زیر I_w را حساب می کند:

$$I_w = S_2 \oplus G(E_{K_E}^{-1}(S_3)) \qquad (۶)$$

به این ترتیب کارگزار با ارسال فایل هایی که شناسه آن ها در I_w وجود دارد به کاربر عمل جستجو را به پایان می رساند.

این طرح دارای دو محدودیت اساسی است:

۱- الگوریتم ذخیره سازی افزونه ها و همچنین الگوریتم جستجو به صورت تعاملی بین کاربر و کارگزار انجام می شوند.

۲- الگوریتم ذخیره سازی افزونه ها پنهای باند زیادی را اشغال میکند به عبارتی چون U_w هم اندازه با I_w است پهنای باند موردنیاز دوبرابر می شود.

۱.۱.۲ امنیت اثبات پذیر

به طور کلی امنیت یک سیستم رمزگذاری طی انجام یک بازی بین چالشگر و مهاجم اثبات می شود و با نشان دادن ناچیز بودن احتمال برد مهاجم می توان گفت امنیت سیستم اثبات شده است. در زیر یک نمونه ساده از اثبات امنیت را به اختصار توضیح می دهیم: فرض کنید A مهاجم و C چالشگر باشند . چالشگر برای به چالش کشیدن مهاجم ابتدا کلیدواژه ای را به صورت تصادفی و با طول λ به صوررت $K \in_R \{0,1\}^{|\lambda|}$ تولید می کند و مقدار λ را برای مهاجم می فرستد . مهاجم A با انتخاب کلید واژه های W_0 و W_1 از چالشگر می خواهد که یکی را به انتخاب خود رمز کرده و به مهاجم بازگرداند

یعنی :

$$b \in_R \{0,1\}, C_b = Enc_K(W_b) \quad (۷)$$

با فرض اینکه مهاجم دریچه T_w را در اختیار داشته باشد عمل جستجو را برای کلیدواژه C_b انجام داده و حاصل جستجو را تعیین می کند:

$$search(T_{W_0}, C_b) = b' \quad (۸)$$

مهاجم A ، b' را برای چالشگر می فرستد و چالشگر در صورتی که $b = b'$ باشد مهاجم را برنده اعلام می کند. در امنیت اثبات پذیر هدف نشان دادن ناچیز بودن احتمال برد مهاجم است.

۳ نتیجه گیری و کارهای آینده

در این مقاله مروری بر طرح های موجود برای رمزگذاری جستجوپذیر متقارن داشتیم و طرح های مختلف را از حیث پیچیدگی ، پویایی و سایر قابلیت ها با هم مقایسه کرده و در نهابت یک طرح نمونه را به تفصیل شرح دادیم.

در تحقیقات آینده توجه به نیاز دنیای واقعی ما را به ارایه طرح های عملی تر و برآوردن نیاز های دنیای امروز رمزنگاری نزدیک تر می کند . به عنوان نمونه ارایه طرح های جدید برای جستجو در بین ایمیل ها و یا در سیستم های PHR و هر سامانه چندکاربره دیگری می تواند مفید واقع شود.

همچنین تعمیم روش های رمزگذاری جستجوپذیر برای یافتن گروهی از کلیدواژه ها و یا یافتن واژه های با تکرار زیاد در اسناد رمزشده زمینه مناسبی برای تحقیقات آینده به نظر می رسد.

□□□□□□□□□□

[1] Song, D.X., Wagner, D. and Perrig, A., 2000. Practical Techniques for Searches on Encrypted Data, Proc. of the 2000 IEEE symposium on Security and Privacy (S&P 2000).

[2] Boneh, D., Di Crescenzo, G., Ostrovsky, R. and Persiano, G., 2004, Public key encryption with keyword search, In Advances in Cryptology-Eurocrypt 2004 (pp. 506-522). Springer Berlin Heidelberg.

[3] Van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P. and Jonker, W., 2010. Computationally Efficient Searchable Symmetric Encryption, *EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)*

[4] Bösch, C., Hartel, P., Jonker, W. and Peter, A., 2014. A Survey of Provably Secure Searchable Encryption, ACM Computing Surveys (CSUR), 47(2), p.18.

[5] Goh, E.J., 2003. Secure Indexes , IACR Cryptology ePrint Archive, 2003, p.216.

[6] Curtmola, R., Garay, J., Kamara, S. and Ostrovsky, R., 2006, October. Searchable symmetric encryption: improved definitions and efficient constructions, In Proceedings of the 13th ACM conference on Computer and communications security (pp. 79-88). ACM .

[7] Kamara, S., Papamanthou, C. and Roeder, T., 2012, October. Dynamic searchable symmetric encryption , In Proceedings of the 2012 ACM conference on Computer and communications security (pp. 965-976). ACM.

[8] Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M.C. and Steiner, M., 2014, October Dynamic searchable encryption in very-large databases: Data structures and implementation. In Network and Distributed System Security Symposium (NDSS'14). Vancouver